Fachhochschule Wedel

Diplomarbeit

in der Fachrichtung
Physikalische Technik

MESSPLATZ ZUR BESTIMMUNG DER KOHÄRENZLÄNGE VON LASERSTRAHLUNG

Eingereicht von:	Udo Becker Breiter Weg 98 22880 Wedel Tel. 04103 / 15422

Erarbeitet im:	8. Semester
Abgegeben am :	26. August 1997
Referent:	Prof. Dr. Iven Pockrand
Co-Referent:	Prof. Dr. Michael Anders

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

In der vorliegenden Arbeit soll der Aufbau eines Meßplatzes zur Bestimmung der Kohärenzlänge von Laserstrahlung beschrieben werden. Neben der Ausgangsleistung eines Lasers und der Wellenlänge der emittierte Strahlung ist die Kohärenzlänge und die sich aus ihr ergebende Frequenzbandbreite eine wichtige Größe zur Charakterisierung der Strahlungsqualität. Besonders in der Nachrichtenübertragung und der interferometrischen Meßtechnik besteht oft die Forderung nach einer großen Kohärenzlänge der verwendeten Laserstrahlung.

Der hier beschriebene Aufbau bietet die Möglichkeit die Kohärenzlänge von Laserstrahlung zu vermessen. Als Strahlungsquellen können verschiedenen Laser in den Aufbau eingekoppelt werden. Aus dem sich durch das Michelson-Interfereometr erzeugte Interferenzsignal können Rückschlüsse auf die Kohärenzlänge gemacht werden.

Ein Michelson-Interferomters war bereits vorhanden. Hierauf aufbauend wurden die zur Kohärenzmessung notwendigen Komponenten hinzugefügt. Die Meßergebnisse können an einem PC grafisch dargestellt werden. Hierzu wurde ein Meßprogramm entwickelt.

Kapitel 2

Physikalische Grundlagen

Kohärenz beschreibt die zeitliche und räumliche Korrelation der Phasen von Wellen. Hierbei kann zwischen der zeitlichen oder longitudinalen Kohärenz und der räumlichen oder lateralen Kohärenz unterschieden werden. Diese Unterscheidung ist zwar etwas künstlich, kann aber die unterschiedlichen Ursachen dieser Effekte deutlich machen. Zeitliche Kohärenz bezieht sich auf die Spektralverteilung der Lichtquelle und damit direkt auf die begrenzte Kohärenzzeit τ_c. Die räumliche Kohärenz beschreibt Effekte, die mit der größe der Lichtquelle – und dadurch mit der Korrelation von Strahlen, die von unterschiedlichen Orten auf der Oberfläche der Lichtquelle ausgehen – zusammenhängen. Bei Lasern spielt die räumliche Kohärenz nur ein untergeordnete Rolle. Die Effekte, die hier auftreten, sind fast ausschließlich auf zeitliche Kohärenz zurückzuführen, die im Folgenden behandelt werden soll.

2.1 Allgemeines Interferenzgesetz

Man bezeichnet zwei Wellen als kohärent, wenn sie eine zeitlich konstante Phasendifferenz aufweisen. In der Praxis kann diese Bedingung nur nährungsweise erfüllt werden, man spricht dann von Teilkohärenz.

Im Beobachtungspunkt P der Abbildung 2.1 soll die Interferenz von zwei ebenen Teilwellen beobachtet werden, die aus derselben Quelle Q (z.B. Laser) stammem, aber unterschiedliche optische Wege zurückgelegt haben. Die beiden Wellen können durch

$$\underline{E}_1(r,t) = \underline{\hat{E}}_1 e^{j(\omega t - \vec{k}_1 \cdot \vec{r} + \varphi_1)} \qquad \underline{E}_2(r,t) = \underline{\hat{E}}_2 e^{j(\omega t - \vec{k}_2 \cdot \vec{r} + \varphi_2)} \tag{2.1}$$

beschrieben werden. Legt die Welle 2 einen kürzeren Weg zurück, so eilt sie der Welle 1 um einen der Laufzeit τ entsprechenden Phasenwinkel $\delta = \omega\tau$ vor, und man erhält für den – an einem festen Ort interressierenden – zeitabhängigen Anteil der Feldstärken:

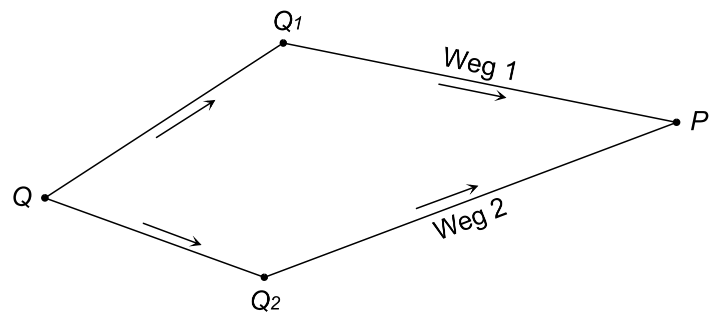

Abbildung 2.1: Interferenz zweier Teilstrahlen die unterschiedliche Wege zurücklegen

$$\underline{E}_1 = \underline{\hat{E}}_1 e^{j\omega t} \qquad \underline{E}_2 = \underline{\hat{E}}_2 e^{j\omega(t+\tau)} \tag{2.2}$$

Damit wird die Feldstärke im Punkt P: $\underline{E}_P = \underline{E}_1 + \underline{E}_2$ und die mittlere Intensität I_P:

$$I_P = \frac{1}{2}\varepsilon c\overline{\underline{E}_P\underline{E}_P^*} = \frac{1}{2}\varepsilon c\overline{(\underline{E}_1 + \underline{E}_2)(\underline{E}_1^* + \underline{E}_2^*)} = \frac{1}{2}\varepsilon c\overline{(|\underline{E}_1|^2 + |\underline{E}_2|^2 + \underline{E}_1\underline{E}_2^* + \underline{E}_1^*\underline{E}_2)} \tag{2.3}$$

Wegen

$$\frac{1}{2}\varepsilon c|\underline{E}_1|^2 = I_1 \qquad \frac{1}{2}\varepsilon c|\underline{E}_2|^2 = I_2 \tag{2.4}$$

$$\underline{E}_1\underline{E}_2^* + \underline{E}_1^*\underline{E}_2 = 2Re(\underline{E}_1\underline{E}_2^*) \tag{2.5}$$

gilt dann

$$I_P = I_1 + I_2 + \varepsilon cRe\overline{(\underline{E}_1\underline{E}_2^*)} \tag{2.6}$$

Hierbei sehen I_1, I_2 für die Intensitäten der Einzelstrahlen, der dritte Term beschreibt die Interferenz. Jetzt kann man die Korrelationsfunktion einführen:

$$\underline{\Gamma}_{12}(\tau) = \overline{\underline{E}_1(t)\underline{E}_2^*(t+\tau)} = \frac{1}{T}\int_{-T/2}^{T/2} \underline{E}_1(t)\underline{E}_2^*(t+\tau)dt \tag{2.7}$$

Bei Normierung auf die Effektivwerte der Felder erhält man die normierte Korrelationsfunktion

$$\underline{\gamma}_{12}(\tau) = \frac{\underline{\Gamma}_{12}(\tau)}{\Gamma_{11}(0)\Gamma_{22}(0)} = \varepsilon c\frac{\underline{\Gamma}_{12}(\tau)}{2\sqrt{I_1 I_2}} \tag{2.8}$$

Damit läßt sich sie Intensität im Punkt P (2.6) durch die normierte Korrelationfunktion ausdrücken

$$I_P = I_1 + I_2 + 2\sqrt{I_1 I_2} Re(\underline{\gamma}_{12}(\tau)) \quad (2.9)$$

Dies ist das allgemeine Interferenzgesetz für teilkohärentes Licht. Die normierte Korrelationsfunktion $\gamma_{12}(\tau)$, der wesentliche Teil des Interferenzterms, ist abhängig von der Laufzeitdifferenz τ und damit der Lage des Punktes P. Die Laufzeitunterschiede müssen zur Erzielung einer hohen Kohärenz klein gegen die mittlere Emissionszeit τ_0 sein. Für $\underline{E}_1 = \underline{E}_2$ und $\tau = 0$ ist $\gamma = 1$ und beide Teilwellen sind in Phase. Man hat also immer konstruktive Interferenz.

Die Qualität der mit einem Interferometer hergestellten Streifenmuster kann quantitativ mit der Sichtbarkeit beschrieben werden, die zuerst von Michelson formuliert wurde. Sie kann auch als Modulation M oder Kontrastfunktion bezeichnet werden.

$$M = \frac{I_{max} - I_{min}}{I_{max} + I_{min}} \quad (2.10)$$

Hier sind I_{max} und I_{min} die Intensitäten, die einem Maximum und angrenzenden Minimum im Interferenzmuster entsprechen.

2.2 Kohärenzfunktion eines Gaslasers

Wenn ein Atom oder ein Molekül in eine Gaslaser Strahlung emittiert entsteht eine schmale, quasimonochromatische Linie der Frequenz ν_0. Bewegt sich das Atom mit der relativen Geschwindigkeit v auf eine Betrachter zu oder von ihm weg, so wird dieser eine Frequenz

$$\nu = \nu_0 \left(1 \pm \frac{v}{c}\right) \quad (2.11)$$

wahrnehmen. Diesen Effekt nennt man den Dopplereffekt nach Christian Doppler (1803 - 1853), der die Frequenzverschiebung 1842 für Schallwellen erklärte. Befinden sich Atome im thermischen Gleichgewicht, so sagt die statistische Mechanik, daß sich eine Maxwellsche Geschwindigkeitsverteilung ergibt. Die spektrale Verteilungsfunktion, die sich aus der Bewegung der Atome ergibt, hat die Form [1]

$$P(\omega) = \sqrt{\pi}\tau_0 e^{-(\omega-\omega_0)^2(\tau_0/2)^2} \quad (2.12)$$

Es ergibt sich eine Gaussche Linie bei der Frequenz ω_0 mit einer Breite von

$$\Delta\omega = \frac{1}{\tau_0} \quad (2.13)$$

Betrachtet man eine Folge von emittierten Wellenzügen, so variiert Ihre Dauer um den Mittelwert τ_0. Die Phasenbeziehungen sind zufällig (Abbildung 2.2).

[1] Robert Guenther, Modern Optics [3], S. 300

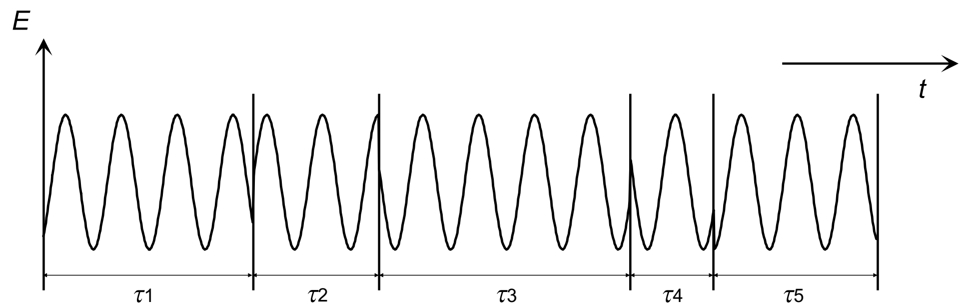

Abbildung 2.2: Folge von Wellenzügen unterschiedlicher Lebensdauer τ

Will man die Frequenzabhängigkeit der Intensität ausdrücken, so läßt sich für die Interferenz zweier Teilstrahlen gleicher Intensität schreiben

$$I_P = \int_0^\infty I(\omega)(1 + \cos\omega\tau)d\omega \tag{2.14}$$

Das Integral enthält einen konstanten und einen oszillierenden Term. Der konstante Term ist

$$I_0 = \int_0^\infty I(\omega)d\omega \tag{2.15}$$

und der oszillierende Term ist

$$\int_0^\infty I(\omega)\cos\omega\tau\, d\omega \tag{2.16}$$

Jetzt läßt sich eine normierte Intensitätsverteilung definieren. Dies ist eine spektrale Verteilungsfunktion.

$$P(\omega) = \frac{I(\omega)}{\int_0^\infty I(\omega)d\omega} = \frac{I(\omega)}{I_0} \tag{2.17}$$

Durch eine Fouriertransformation der Gausschen Verteilungsfunktion für Atom- oder Molekularemission (2.12) kann $Re(\underline{\gamma}_{12}(\tau))$ berechnet werden.

$$Re(\underline{\gamma}_{12}(\tau)) = \int_0^\infty P(\omega)\cos\omega\tau\, d\omega \tag{2.18}$$

Das Resultat ist wieder eine Gaussche Kurve

$$Re(\underline{\gamma}_{12}(\tau)) = e^{-(\tau/\tau_0)^2}\cos\omega_0\tau \tag{2.19}$$

$$Re(\underline{\gamma}_{12}(\tau)) = \gamma_{12}(\tau)\cos\omega\tau \tag{2.20}$$

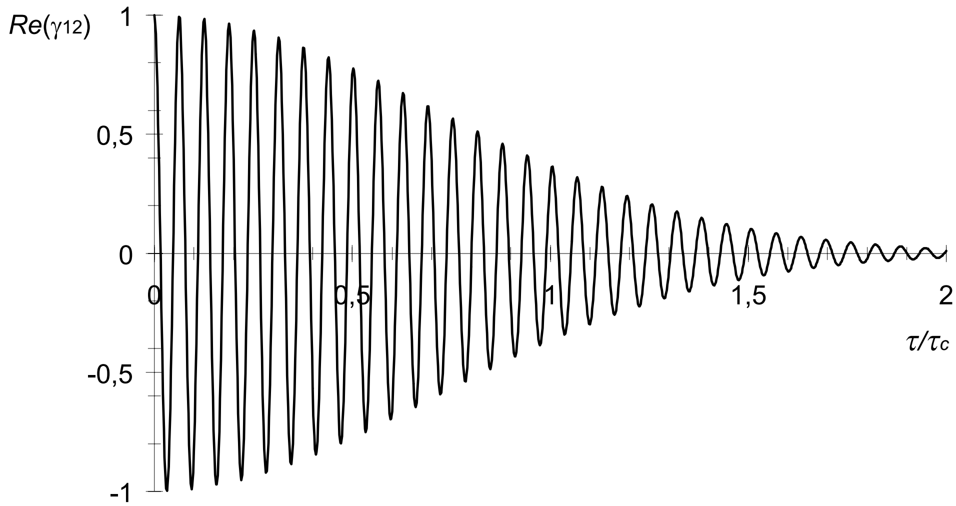

Abbildung 2.3: Korrelationsfunktion

Die Korrelationfunktion besteht also aus einem schnell variierenden Term $\cos \omega_0 \tau$ und einem langsamer variierenden Term $e^{-(\tau/\tau_0)^2}$, der die Amplitude des schnell variierenden Terms moduliert. Wenn $\tau \ll \tau_0$ kommt nur der schnell variierende Term zum tragen.

$$Re(\underline{\gamma}_{12}(\tau)) \approx \cos \omega_0 \tau \tag{2.21}$$

Hier erhält man das Ergebnis für eine monochromatische Welle, also ideal kohärente Strahlung. Kommt τ jedoch in die Größenordnung von τ_0, so kann der Effekt der Frequenzverschiebung, ausgedrückt in (2.12), in dem langsamer variierenden Term beobachtet werden. Dieser Term wird im Folgenden als Kohärenzgrad bzw. Kohärenzfunktion bezeichnet.

$$\gamma_{12}(\tau) = e^{-(\tau/\tau_0)^2} \tag{2.22}$$

Die im Punkt P auftretende Intensität des Interferenzmusters kann jetzt in Abhängigkeit von der Verzögerung τ ausgedrückt werden.

$$I_P(\tau) = I_1 + I_2 + 2\sqrt{I_1 I_2}\gamma_{12}(\tau) \cos \omega_0 \tau \tag{2.23}$$

Die maximal auftretende Intensität beträgt

$$I_{max} = I_1 + I_2 + 2\sqrt{I_1 I_2}\gamma_{12}(\tau) \tag{2.24}$$

und die Minimalintensität ist

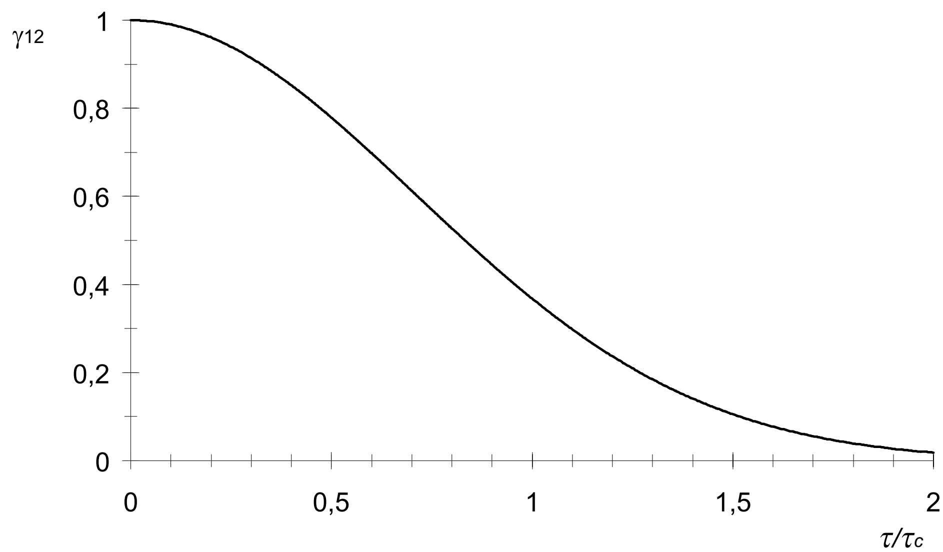

Abbildung 2.4: Kohärenzfunktion

$$I_{min} = I_1 + I_2 - 2\sqrt{I_1 I_2}\gamma_{12}(\tau) \tag{2.25}$$

Über die Kontrastfunktion M (2.10) besteht jetzt eine Möglichkeit, $\gamma_{12}(\tau)$ zu messen

$$M = \frac{I_{max} - I_{min}}{I_{max} + I_{min}} = \frac{2\sqrt{I_1 I_2}}{I_1 + I_2}\gamma_{12}(\tau) \tag{2.26}$$

Bei gleichen Intensitäten stimmt also der Kohärenzgrad γ_{12} mit der Kontrastfunktion M berein:

$$M = |\gamma_{12}(\tau)| \tag{2.27}$$

Damit ergeben sich bei der Interferenz für Intensität und Kontrast die Spezialfälle in Tabelle 2.1.

	für gleiche Intesitäten:
inkohärenter Grenzfall: $\tau_0 \to 0$ bzw. $\tau \gg \tau_0$; $\gamma_{12} = 0$	
$I_P = I_1 + I_2$	$I_P = 2I_0$
$M = 0$, da $I_{max} = I_{min} = I_P$	$M = 0$
kohärenter Grenzfall: $\tau_0 \to \infty$ bzw. $\tau \ll \tau_0$; $\gamma_{12} = 1$	
$I_P = I_1 + I_2 + 2\sqrt{I_1 I_2} \cos \omega\tau$	$I_P = 2I_0(1 + \cos \omega_0\tau)$
$I_{max} = I_1 + I_2 + 2\sqrt{I_1 I_2}$	$I_{max} = 4I_0$
$I_{min} = I_1 + I_2 - 2\sqrt{I_1 I_2}$	$I_{min} = 0$
$M = \frac{2\sqrt{I_1 I_2}}{I_1 + I_2}$	$M = 1$
Teilkohärenz: $0 < \gamma_{12} < 1$	
$I_P = I_1 + I_2 + 2\sqrt{I_1 I_2}\gamma_{12} \cos \omega_0\tau$	$I_P = 2I_0(1 + \gamma_{12} \cos \omega_0\tau)$
$I_{max} = I_1 + I_2 + 2\sqrt{I_1 I_2}\gamma_{12}$	$I_{max} = 2I_0(1 + \gamma_{12})$
$I_{min} = I_1 + I_2 + 2\sqrt{I_1 I_2}\gamma_{12}$	$I_{min} = 2I_0(1 - \gamma_{12})$
$M = \frac{2\sqrt{I_1 I_2}}{I_1 + I_2}\gamma_{12}$	$M = \gamma_{12}$

Tabelle 2.1: Spezialfälle der Interferenz für Intensität und Kontrast

2.3 Kohärenzzeit und -länge

Über eine Definition der Kohärenzzeit herrscht in der Literatur keine Einigkeit. Für praktische Zwecke ist folgende Definition durch die Breite des Frequenzspektrums $\Delta\omega$ meist ausreichend.

$$\tau_c = \frac{1}{\Delta\omega} = \tau_0 \tag{2.28}$$

Nach dieser Definition ergibt sich bei einer der Kohärenzzeit entprechenden Verzögerung $\gamma_{12} = 1/e$ (vergleiche (2.22)). Der Kohärenzzeit entspricht eine Kohärenzlänge von

$$\ell_c = c\tau_c = \frac{c}{\Delta\nu} \tag{2.29}$$

Bei relativen kurzen Kohärenzlängen in der Größenordnung der Wellenlänge λ ist es manchmal sinnvoll, die Kohärenzlänge ℓ_c in Bezug zur Wellenlänge auszudrücken.

$$\frac{\Delta\nu}{\nu} = \frac{|\Delta\lambda|}{\lambda} \tag{2.30}$$

$$\ell_c = \frac{\lambda^2}{\Delta\lambda} \tag{2.31}$$

Über die Kohärenzlänge kann also auch die spekrale Breite bestimmt werden.

2.4 Michelson-Interferometer

In Abbildung 2.5 ist der Strahlengang im Michelson-Interferometer prinzipiell dargestellt.

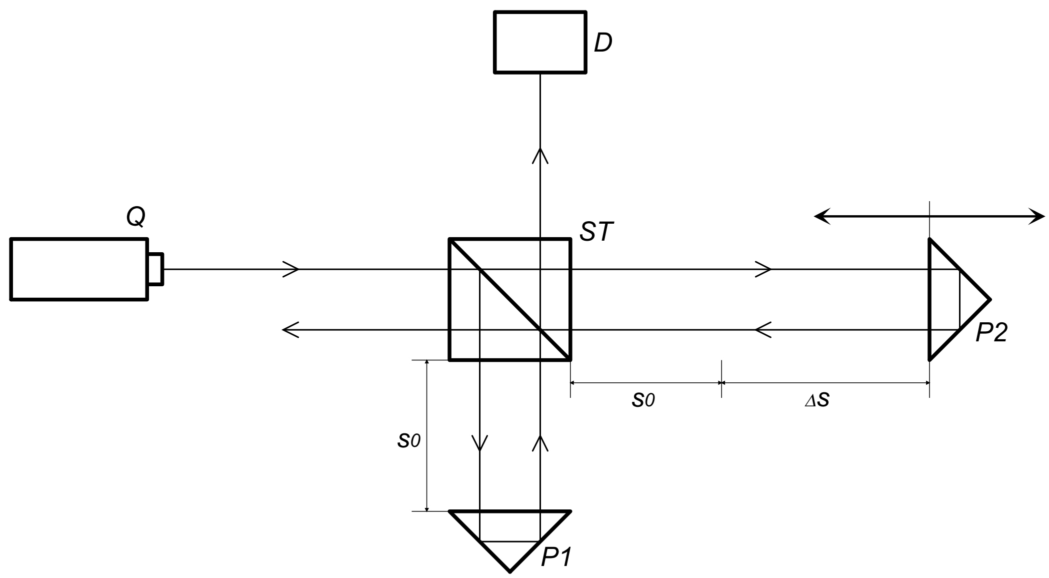

Abbildung 2.5: Strahlengang beim Michelson-Interferometer

Die von einer Lichtquelle *Q* (z.B. Laser) ausgehende Strahlung trifft auf einen Strahlteiler *ST* und wird in zwei senkrecht zueinander stehende Anteile idealerweise gleicher Intensität zerlegt. Der Referenzstrahl wird in einem Umlenkprisma *P1* durch Totalreflektion zweimal um jeweils 90^o abgelenkt und somit zum Strahlteiler *ST* zurückgeworfen. Hierbei legt er eine feste Wegstrecke $2s_0$ zurück. Der zweite Anteil, der Objektstrahl, wird ebenfalls durch ein Umlenkprisma *P2* zurückgeworfen, die Entfernung dieses Prismas vom Strahlteiler ist jedoch variabel. Die Wegstrecke ist hier um den Betrag $2\Delta s$ länger, es ergibt sich eine Verzögerung

$$\tau = \frac{2\Delta s}{c} = \frac{d}{c} \tag{2.32}$$

Mit dem Michelson-Interferometer kann nicht nur die Intensität an einem Punkt beobachtet werden, man hat die Möglichkeit die Intensitätsverteilung über ein Strahlprofil zu sehen. Nun ist es im allgemeinen nicht so, daß über den gesamten Strahlquerschnitt ein exakt gleicher Abstand Δs vorliegt. Eine relativ kleine Änderung des Abstandes δ von einer halben Wellenlänge wird einen vollständigen Cosinus der Modulation sichtbar machen. Der Unterschied im Abstand kann in der Praxis durch die Justierung des Objektprismas auf mehrere Wellenlängen eingestellt werden, so daß einige Maxima und Minima der Intensität sichtbar werden. Da dieser unterschiedliche Abstand im Bereich einiger Mikrometer sehr klein in Bezug

zur Kohärenzlänge üblicher Laser ist kann man davon ausgehen, daß sich die Werte für I_{max} und I_{min} nur unwesentlich ändern. Somit kann man für einen Abstand sowohl Werte für die Maximale als auch für die Minimale Intensität feststellen. In Abbildung 2.6 sind für drei verschiedene Kohärenzgrade die Interferenzstreifenmuster bei gleichen Teilstrahlintensitäten I_1 und I_2 dargestellt.

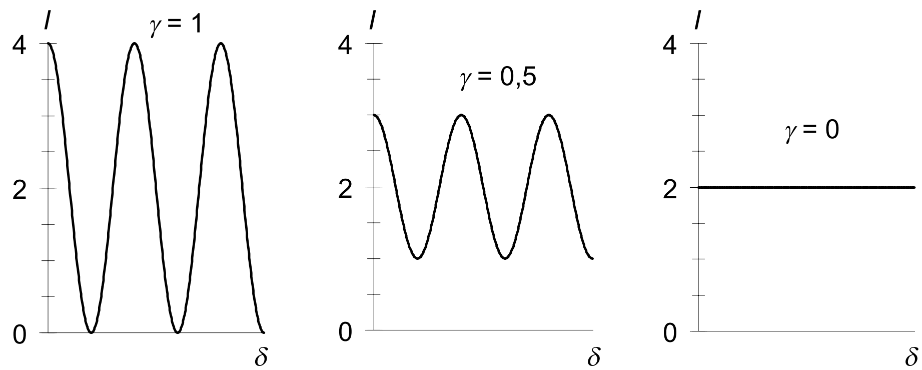

Abbildung 2.6: Interferenzstreifenmuster beim Michelson-Interferometer

Kapitel 3

Beschreibung der Meßanlage

Der von mir verwendete Interferometer-Aufbau ist im Rahmen einer Diplomarbeit von Ralf Lindner [8] entstanden. In dieser Arbeit wurde mit Hilfe eines Michelson-Interferometers ein interferometrischer Längenmeßplatz realisiert. Dieser Aufbau konnte von mir weitgehend übernommen werden, wobei die interferometrische Längenmessung ebenfalls noch durchgeführt werden kann. Der Aufbau ist in der oben genannten Arbeit ausführlich beschreiben. In diesem Kapitel werden die für die Kohärenzmessung relevanten und die neu hinzugefügten Kompenenten behandelt und der Meßvorgang erklärt.

3.1 Gesamtkonzept

In Abbildung 3.1 ist der Interferometer-Aufbau schematisch dargestellt. Über den PC kann der Motor *M* angesteuert werden, der eine Spindel antreibt. Durch die Spindel wird der Wagen *W* mit dem Objektprisma *P*2 auf der Linearführung *L* bewegt. Hier lassen sich Wegdifferenzen zwischen 0 und 90 cm einstellen. Der Strahl des zu vermessenden Lasers wird über einen Umlenkspiegel *SP* in den Aufbau eingekoppelt und trifft auf einen Strahlteilerwürfel *ST*. Hier wird der Strahl in zwei Teilstrahlen aufgeteilt, die vom Referenzprisma *P*1 und vom Objektprisma *P*2 zurückgeworfen werden und wiederum den Strahlteiler durchlaufen. Der Objektstrahl hat eine längere Wegstrecke zurückgelegt, in der Überlagerung der beiden Strahlen hat dieser Anteil also eine Verzögerung τ erfahren deren Einfluß auf das Interferenzmuster untersucht werden soll. Die Überlagerung der beiden Strahlen wird durch ein Mikroskop-Objektiv *MO* aufgeweitet und durchläuft einen Tubus *T*, bevor sie auf den Detektor – eine CCD-Zeilenkamera – trifft.

Das detektierte Signal wird durch einen Subtrahierer *SUB* von einem Offsetanteil befreit. Hierauf wird weiter unten eingegangen.

Am Oszilloskop kann die Intensitätsverteilung auf der Kamerazeile betrachtet werden. Im Oszilloskop erfolgt eine Digitalisierung. Das digitale Signal wird über eine serielle Schnittstelle in den PC eigelesen. Es werden ebenfalls durch Abblenden der entsprechenden Strahlen die Intensitätsverteilungen der beiden Teil-

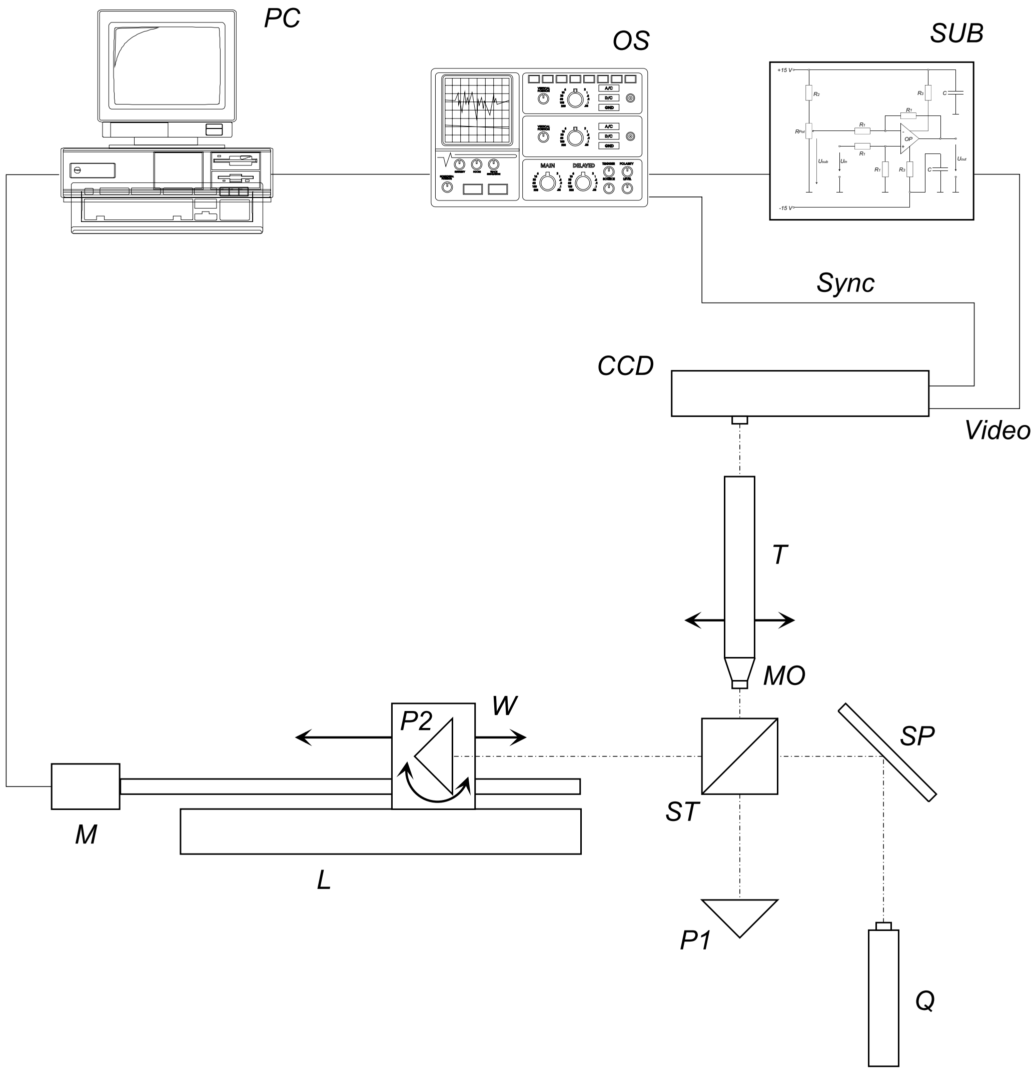

Abbildung 3.1: Interferometer-Aufbau

strahlen und eine Dunkelmessung aufgenommen. Vom Meßprogramm wird der Kohärenzgrad für diese Wegdifferenz ermittelt. Führt man mehrere Messungen für verschiedene Wegdifferenzen durch kann hieraus die Kohärenzfunktion und die Kohärenzlänge berechnet werden.

Von mir wurden einige kleine Änderungen am Aufbau vorgenommen. Die Halterung für das Prisma, das den Objektstrahl umlenkt, wurde verändert. Hierdurch wird die Einkopplung des Objektstrahls in das Mikroskop-Objektiv erleichtert. Durch eine neue Positionierung des Referenzprismas lassen sich gleiche Weglängen für beide Teilstrahlen realisieren. Die Beobachtungsoptik wurde mit einem Linearversteller versehen, somit kann die Position des Mikroskop-Objektivs bei neuer Justierung angepaßt werden. Auf der Linearführung des Wagens wurde eine Maßzeile angebracht, von der direkt die eingestellte Wegdifferenz abgelesen werden kann.

3.2 CCD-Kamera

In dem Interferometer-Aufbau waren als Detektoren zwei Photodioden vorhanden, deren Signal gegen ein Schwelle geprüft wurde. Die Messung erfolgte bei bewegtem Wagen, es wurde eine Zählung der Interferenzstreifen durchgeführt. Für die Aufnahme eines Interferenzmusters an einer festen Position, insbesondere wenn mehrere Messungen an der gleichen Position durchgeführt werden sollen, kann ein Zeilensensor eingesetzt werden. Dieser Sensor liefert als Signal die Intensitätsverteilung eines Schnittes durch das Strahlprofil.

Eine eindimensionale CCD-Kamera CCD 151 der Firma Fairchild Weston mit Entwicklungsboard war bereits aus einer Diplomarbeit von Stephan Schulz [9] vorhanden. Diese CCD-Kamera ist mit einer internen Taktung versehen und liefert ein Videosignal, das direkt an ein Oszilloskop angeschlossen werden kann.

Die Sensorzeile ist aus $N = 3456$ Einzelpixeln aufgebaut. Die Breite der Pixel beträgt $d_{Pixel} = 7 \mu m$. Über ein Potentiometer läßt sich die Pixelfrequenz in einem Bereich von 280 kHz bis 2,3 MHz einstellen. Laut Datenblatt ist die Kamera mit einer Betriebsspannung von +17 Volt zu betreiben. Durch das Netzteil der interferometrischen Längenmessung steht u.a. eine Spannung von +15 Volt zur Verfügung. Im Betrieb mit dieser Spannung war kein Unterschied im Ausgangssignal zu einem Betrieb mit einem Labornetzteil mit +17 Volt festzustellen. Die Spannungsversorgung erfolgt deswegen durch o.g. Netzteil.

Bei der Meßwerteaufnahme mit dem Oszilloskop muß darauf geachtet werden, daß das Oszilloskop im DC-Modus betrieben wird. Bei Messungen im AC-Modus wären verschiedene Messungen, z.B. die Aufnahme eines Interferenzmusters und einer Nullintensität, nicht vergleichbar, da die Nullinie verschoben wird. Die CCD-Kamera liefert ein Videosignal mit einem Offset-Anteil von ca. 5 V und daraufliegendem Signal von maximal 2 V. Da eine Darstellung dieses Signals im DC-Modus eine geringe Auflösung im interressierenden Bereich ab 5 V liefert ist es sinvoll, den Offsetanteil vor der Signaleingabe in das Oszilloskop abzuziehen. Aus diesem Zweck wurde eine Subtrahierenschaltung vor dem Oszilloskop zwischengeschaltet.

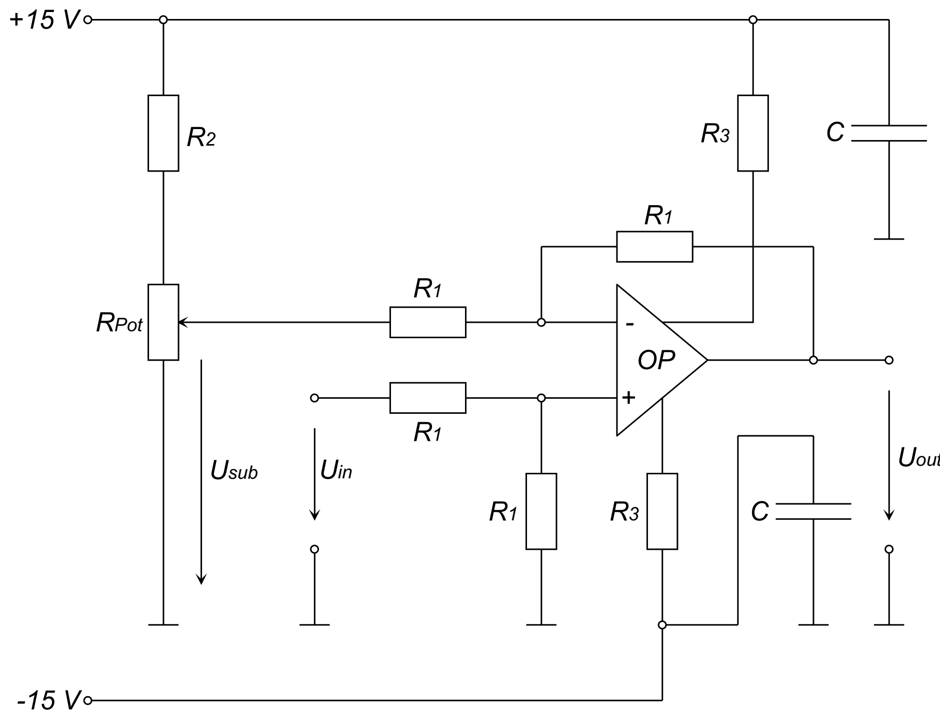

Abbildung 3.2: Subtrahierer-Schaltung

R_1	$100\ k\Omega$
R_2	$2,7\ k\Omega$
R_{Pot}	$0 \cdots 5\ k\Omega$
R_3	$100\ \Omega$
C	$100\ nF$
OP	CA 3240

Tabelle 3.1: Bauelemente der Subtrahierer-Schaltung

Diese Schaltung ist im Gehäuse der Elektronischen Komponenten der interferometrischen Längenmessung integriert und wird mit einer Vorsorgungsspannung von ± 15 V betrieben. Die Betriebsspannung des Operationsverstärkers ist über jeweils einen Widerstand R_3 und einen Kondensator C gepuffert. Hierdurch wird ein Einschwingen des Operationsverstärkers unterdrückt. In der Subtrahiererschaltung wurden vier gleiche Widerstände R_1 verwendet, so daß die Verstärkung zu 1 wird. Die vom Meßsignal abzuziehende Spannung U_{sub} kann über das Potentiometer R_{Pot} auf Werte zwischen 0 und 9,7 Volt eingestellt werden, der einzustellende Wert liegt bei ca. 5 Volt. Das Ausgangssignal der Schaltung ergibt sich zu

$$U_{out} = U_{in} - U_{sub} \tag{3.1}$$

Der Offsetanteil des Videosignals ändert seinen Betrag jedoch mit der Größe des Meßsignals. Liegt eine Bestrahlung des Sensors vor vermindert sich der Offsetanteil in Bezug zu einer Messung mit Nullintensität. Dieses Verschieben der Nullinie kann über die Einstellung der Pixelfrequenz minimiert werden. Bei maximal einstellbarer Pixelfrequenz liegt diese Verschiebung in der Größenordnung des Auflösungsvermögens des Oszilloskops, als unter 0,5 % der Maximalwerte.

Um die Linearität der CCD-Kamera zu überprüfen wurde sie mit unterschiedlichen Strahlleistungen Φ bestrahlt und die Ausgangsspannung U über das Oszilloskop detektiert. Die Strahlleistung wurde mit einem Photometer gemessen.

Φ [μW]	U [mV]	Φ [μW]	U [mV]
6,4	70	62	750
8,4	95	70	800
9,4	105	81	850
9,8	110	85	950
11	125	98	1100
13	150	115	1200
14	160	120	1300
17,5	175	135	1400
19	200	140	1500
23	250	150	1600
24	280	160	1700
28	320	180	1700
32	380	200	1800
38	450	220	1800
43	500	250	1800
46	580	400	1800
52	620	600	1800
56	680	800	1800

Tabelle 3.2: Linearität der CCD-Kamera

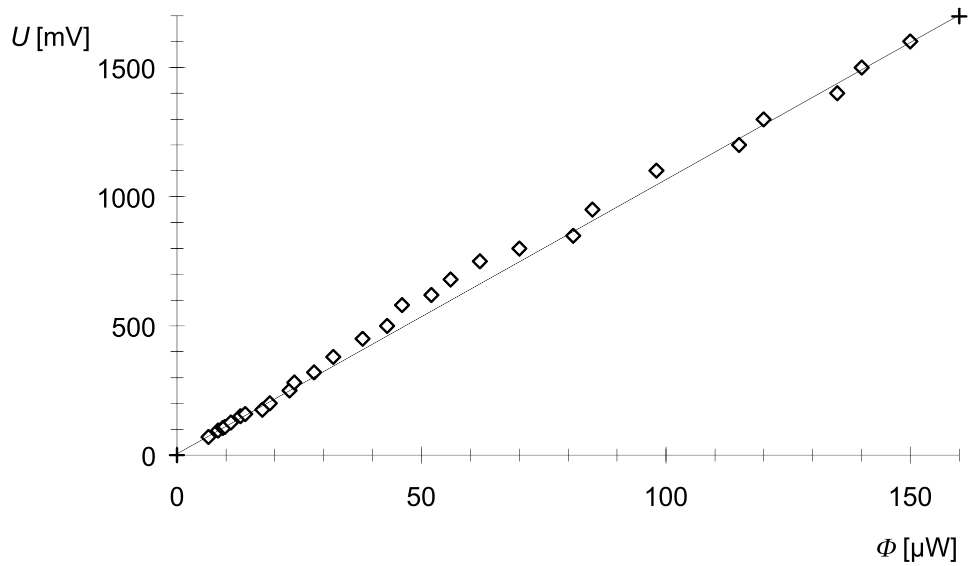

Abbildung 3.3: Linearität der CCD-Kamera

Wie in Abbildung 3.3 erkennbar besteht ein linearer Zusammenhang zwischen dem Ausgangssignal und der eingestrahlten Leistung (die proportional zur Intensität ist). Anhand der Meßwerte läßt sich auch das maximale Ausgangssignal von ca. 1,7 Volt ablesen.

Das Ausgangssignal der Kamera ist bei der Messung eines Interferenzmusters von einem starkem Rauschen überlagert wodurch eine genaue Bestimmung der Maximal- und Minimalwerte nicht möglich ist. Dies ist jedoch erforderlich, da diese Werte für die Berechnung der Modulation benötigt werden.

Zu Beginn meiner Arbeit habe ich dieses Rauschen mit einer elektronischen Tiefpasschaltung herausgefiltert. Hierzu habe ich eine Tiefpass 4. Ordnung mit 4 Operationsverstärkern aufgebaut. Dies führte jedoch zu Problemen, da sich die Frequenz des Rauschens sich mit der Frequenz des Messignals ändert. Dieses dynamische Rauschen ist hauptsächlich auf Interferenzen an der Glasabdeckung der CCD-Zeile zurückzuführen, die fest mit der Zeile verbunden ist und nicht ohne Beschädigung dieser entfernt werden kann. Aus diesem Grund habe ich einen dynamischen Filter als Software in das Meßprogramm integriert. Dieser Filter ist in Abschnitt 3.5.3.1 beschreiben.

Ein weiteres Problem mit der Teifpasschaltung ergab sich dadurch, daß der Offsetanteil des Videosignals sich stärker ändert wenn in der nachfolgenden Elektronik Kapazitäten verwendet werden.

Neben dem Videosignal liefert die CCD-Kamera einen Synchronisations-Impuls zu Beginn jeder Auslesung der Sensorzeile. Dieses Signal wird auf den zweiten Eingang des Oszilloskops gegeben und dient zur Triggerung.

3.3 Oszilloskop

Als Schnittstelle zwischen CCD-Zeile und PC steht ein Digital Speicher Oszilloskop vom Typ Phillips PM3350A zur Verfügung. Diese Oszilloskop liefert ein digitales Signal mit einer Auflösung von 1024 x 256 Punkten. Ein digital gespeichertes Signal läßt sich vom Oszilloskop über eine serielle Schnittstelle (RS-232C) direkt auf einem Plotter oder Drucker ausgeben.

Die Digitalisierung der Meßdaten wollte ich gerne durch eine A/D-Wandler-Steckkarte im Meßcomputer durchführen lassen. Dies hätte folgende Vorteile: Auf das Oszillosokop könnte verzichtet werden, die Datenübertragung von der CCD-Kamera zum PC wäre wesentlich schneller und das Meßsignal könnte mit einer höheren Auflösung detektiert werden. Zur Synchronisation von CCD-Kamera und A/D-Wandler müßte das interne Taktsignal der Kamera durch eine externe Taktung vom PC ersetzt werden. Bei Versuchen mit einer preisgünstigen A/D-Wandler-Steckkarte stellte sich jedoch heraus, daß mit den vorhandenen Komponenten keine Kompatibilität in den Taktraten erreicht werden kann. Wird die Pixelfrequenz der CCD-Kamera vom PC durch daß Meßprogramm vorgegeben kann keine ausreichende Geschwindigkeit erreicht werden. Die einzelnen Pixel werden zu langsam ausgelesen, die Integrationszeit wird also zu groß und das Signal wird unbrauchbar. Dies könnte durch einen wesentlich schnelleren PC umgangen werden. Hier muß dann jedoch auch die A/D-Wandler-Steckkarte in sehr kurzer Zeit die Digitalisierung durchführen. Es müßte also eine hochwertige – und somit auch teure – Karte angeschafft werden.

Aus diesen Gründen wird die Digitalisierung mit dem Oszilloskop durchgeführt. Der Vorteil am Einsatz eines Oszilloskops ist, daß das Signal in Echtzeit betrachtet werden kann und somit eine Justierungshilfe zur Verfügung steht.

3.4 Meßcomputer

Der Messcomputer ist ein IBM-Kompatibler PC mit 80386-Prozessor und einer Taktfrequenz von 21 MHz. Über eine serielle Schnittstelle wird das Plotsignal vom Oszilloskop eingelesen. Man erhält für jede Messung eine plotbare Datei, deren Daten vom Meßprogramm in X-Y-Wertepaare transformiert werden. Als Format der Plotdatei wurde wurde ein HPGL-kompatibles Protokoll für einen digitalen Plotter vom Typ HP7550 gewählt. Dieser Plotter steht im Laserlabor zur Verfügung, somit besteht die Möglichkeit bereits auf Festplatte gespeicherte Messungen auszuplotten.

Die Signale zur Motorsteuerung werden durch eine A-1210 I/O-Steckkarte der Firma MessComp übertragen. Die Belegung der einzelnen Kanäle wurde aus der Arbeit von Ralf Lindner [8] übernommen da ebenfalls die Daten zur interferometrische Längenmessung über diese Karte ein- und ausgegeben werden.

3.5 Durchführung der Messung

3.5.1 Vorbereitende Schritte

Je nach zu vermessendem Laser sollte dieser vor Beginn der Messung eine Zeit von einer Stunde (Ar^+-Laser) bis zu einem Tag (HeNe-Laser) warmlaufen, erst dann emittiert der Laser stabil Strahlung und die zeitlichen Intensitätsschwankungen nehmen ab.

Vor Beginn der Messung ist die Verkabelung der Geräte zu überprüfen. Neben den Spannungsversorgungen sind die Signalleitungen wie folgt angeordnet:

- BNC-Kabel vom Video-Ausgang der CCD-Kamera zum Eingang des Subtrahierers.
- BNC-Kabel vom Ausgang des Subtrahierers zum Eingang Kanal A am Oszilloskop.
- BNC-Kabel vom Sync-Ausgang der CCD-Kamera zum Eingang Kanal B am Oszilloskop.
- Serielles Plotterkabel vom Oszilloskop zum PC.

Das Oszilloskop muß eingeschaltet werden, es benötigt jedoch keine Warmlaufzeit. Folgende Einstellungen am Oszilloskop müssen zur Datenübertragung zum PC geändert werden:

- Durch drücken der Taste `DIGITAL MEMORY` den digitalen Speicherbetrieb einstellen.
- Mit der Taste `TRIG or X SOURCE` den Trigger auf Kanal B einstellen. An diesem liegt der Sync-Impuls an. Im Display erscheint ein B.
- Durch gleichzeitiges drücken der Tasten `MENU` und `AUTO SET` das Service-Menu aufrufen. Hier müssen im Anwendungswahl-Menu (Softkey `APPL`) folgende drei Parameter verändert werden:
  - Untermenu `RS232` aufrufen und hier Untermenu `BAUDRATE` aufrufen. Den Parameter `OUT-SPD` (Ausgangs-Geschwindigkeit) auf `9600` setzen. Danach zweimaliges drücken des Softkeys `RETURN`.
  - Mit Softkey `PLOT` das Plotmenu aufrufen. Mit `PLOT-KEY` die Angabe `PLOT DIGITAL` einstellen.
  - Durch drücken von `TYPE` den Plottertyp `HP7550` auswählen. Anschließend `RETURN` drücken.

  Mit der Taste `AUTO SET` wird das Service-Menu wieder geschlossen.

Leider können diese Änderungen nicht gespeichert werden, so daß die Einstellungen nach jedem Einschalten des Oszilloskops vorgenommen werden müssen.

Der Meßcomputer sollte gestartet werden, da über das Meßprogramm die Motorsteuerung des Wagen erfolgt und dieser zur Justierung des Aufbaus bewegt werden muß.

3.5.2 Justierung des Aufbaus

Zunächst wird der Laserstrahl über den Umlenkspiegel so in den Interferometer-Aufbau eingekoppelt, daß er parallel zur Linearführung des Wagens verläuft. Der Strahls sollte den Strahlteilerwürfel ungefähr in der Mitte treffen. Die Einstellungen des Strahlteilers und des Referenzprismas müssen in der Regel nicht verändert werden.

Der Tubus mit dem Mikroskop-Objektiv wird durch seine Linearführung so in den Strahlengang geführt, daß über die Verstellmöglichkeiten des Objektprismas auf dem Wagen der Objekstrahl mit dem Referenzstrahl für jede Wagenposition überlagert werden kann und daß die Überlagerung der Strahlen in das Objektiv einfällt. Es ist darauf zu achten, daß beide Teilstrahlen für sich die Kamerazeile möglichst gleichmäßig ausleuchten und daß die auf die Kamera treffenden Intensitäten der Teilstrahlen möglichst gleich hoch sind. Dadurch wird eine eine hohe Modulation erreicht. Die Intensitätsverteilung auf der Kamerazeile kann am Oszilloskop überprüft werden.

Über die Verstellmöglichkeiten am Halter des Objektprismas läßt sich sowohl die Ortsfrequenz der Interferenzstreifen als auch ihre Ausrichtung einstellen. Die Kamera sollte so positioniert werden daß die Interferenzstreifen senkrecht zur Ausrichtung der Zeile auftreffen. Für eine Messung ist am besten ein Interferenzmuster mit ca. 3 bis 5 Interferenzstreifen geeignet. Die Breite der Interferenzstreifen, die auf die Kamera fallen, läßt sich ebenfalls durch die Wahl eines Mikroskop-Objektivs mit anderer Vergrößerung verändern.

Am Oszilloskop muß die Horizontale Zeitbasis-Einstellung mit dem Up-Down-Schalter `TB` auf einen Wert von 50 μs/div oder niedriger eingestellt werden. Hierdurch wird gewährleistet, daß das Meßsignal den gesamten zeitlichen Darstellungsbereich des Oszilloskops ausfüllt. Durch die Einstellung der Triggerverzögerung kann mit dem Up-Down-Schalter `TRIG DEL` das Signal in den Darstellungsbereich geschoben werden.

Der Eingangsabschwächer von Kanal A ist zunächst mit dem Up-Down-Schalter `A` auf den Minimalwert von 2 mV/div zu stellen. Jetzt kann bei abgeblendeten Strahlen über das Potentiometer des Subtrahierers der abzuziehende Offsetanteil so eingestellt werden, daß die Nullinie am unteren Bildrand liegt. Über den Eingangsabschwächer kann dann die vertikale Auflösung für ein Meßsignal möglichst hoch eingestellt werden.

3.5.3 Bestimmung der Intensitäten

Die folgende Beschreibung der Aufnahme von Meßwerten soll an einem Beispiel erfolgen. Die angegebenen Messungen wurden an einem Ar^+-Laser durchgeführt. Dabei wurde der Laser im Multimodebetrieb betrieben und durch einen Frequenzfilter nur eine Spektrallinie um die Wellenlänge 514,5 nm vermessen. Die dargestellten Intensitätsverteilungen wurden bei einer Wegdifferenz von 5 cm aufgenommen.

Da die dem PC durch die Plotdaten übermittelten Werte proportional zur Intensität sind und eine genaue Umrechnung kaum möglich ist werden die Intensitäten im Folgenden nur als dimensionslose Werte angeführt. Diese Vereinfachung ist zwar nicht ganz korrekt, kann aber gemacht werden, da die Absolutbeträge der Intensitäten für die Berechnungen nicht relevant sind. Die Verhältnisse zwischen die Intensitäten der verschiedenen Messungen sind durch die Proportionalität gegeben.

3.5.3.1 Maximale und minimale Intensität eines Interferenzstreifenmusters

Wurde die Justierung des Aufbaus durchgeführt und fällt ein Interferenzmuster auf die Detektorzeile, so kann mit der Messung begonnen werden. Eine Beschreibung der im Meßprogramm vorzunehmenden Schritte findet sich im Benutzerhandbuch im Anhang. Durch drücken der `Plot`-Taste am Oszilloskop werden die Daten zum Rechner übertragen.

Abbildung 3.4 zeigt ein Beispiel einer Intensitätsverteilung des Interferenzstreifenmusters auf der CCD-Zeile.

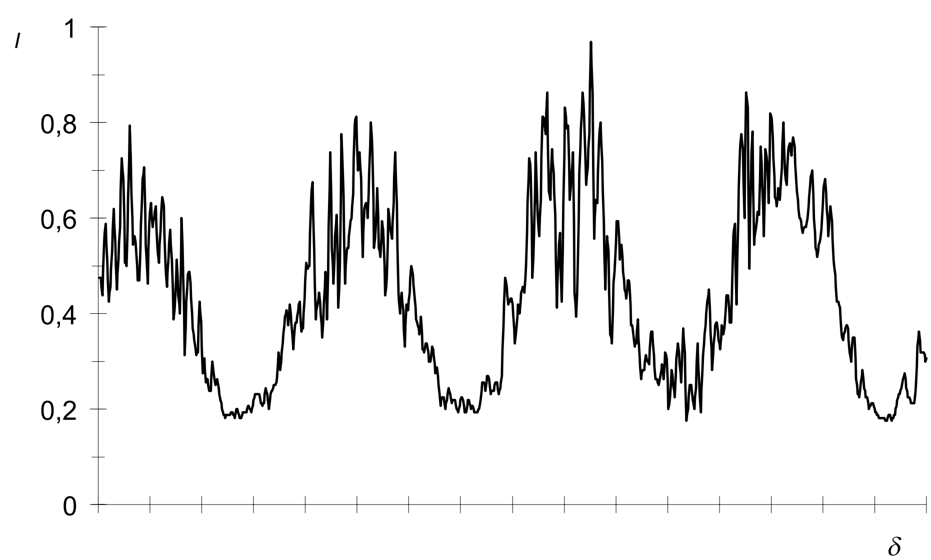

Abbildung 3.4: Gemessene Intensitätsverteilung des Interferenzmusters

Das Interferenzmuster ist von einem starken Rauschen überlagert. Dieses Rauschen soll nun herausgefiltert werden. Im Interferenzmuster sind 4 Maxima und 4 Minima zu erkennen. Dies bedeutet, daß der Wegunterschied über den Meßbreich zwei Wellenlängen oder ungefähr 1 μm beträgt. Für dieses Bild kann eine Signalfrequenz von ungefähr 4 Wellenzügen/Meßbereich definiert werden. Zunächt soll überlegt werden welche Signalfrequenzen auftreten können. Die minimale auswertbare Signalfrequenz liegt bei ungefähr einem Wellenzug/Meßbereich, da hier gerade noch ein Maximum und ein Minimum auftreten. Bei Strahlungsquellen mit geringer Leistung kann durch ein Mikroskop-Objektiv mit kleinerer Aufweitung die auf den Detektor fallende Intensität erhöht werden. Hierdurch werden die detektierten Interferenzstreifen schmaler. Signalfrequenzen von bis zu 20 Wellenzügen/Meßbereich treten dann auf. Noch höhere Signalfrequenzen konnte ich bei meinen Messungen nicht erreichen. Über Horizontale Zeitbasis-Einstellung am Oszilloskop besteht die Möglichkeit, die Signalfrequenz wieder zu senken.

Die Überlegung, bei welcher Frequenz eine Filterfunktion jetzt angreifen sollte kann keinen festen Wert ergeben. Vielmehr muß ein dynamischer Filter zum Einsatz kommen, der seine Grenzfrequenz aus der Signalfrequenz selbst ermittelt. Hier bietet es sich an, mit dem Meßsignal eine Fouriertransformation durchzuführen. Durch die Fourierertranformation wird das Spektrum berechnet. Dies geschieht im Meßprogramm über den Algorithmus der schnellen Foueriertransformation (Fast Fourier Transformation FFT). Abbildung 3.5 zeigt die FFT des Meßsignals.

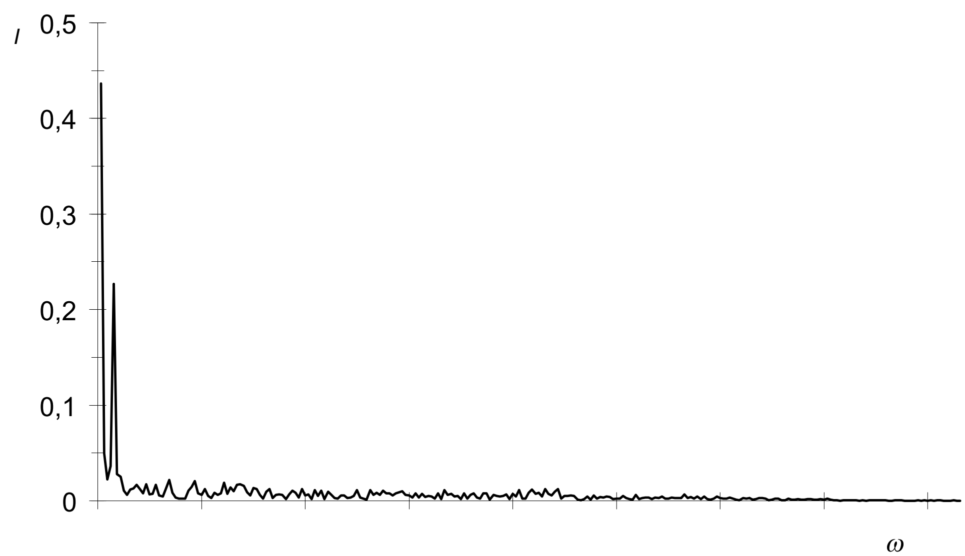

Abbildung 3.5: FFT der Intensitätsverteilung des Interferenzmusters

Aus dem Fourierspektrum des Signals läßt sich jetzt die Signalfrequenz ermit-

teln. Sie ist bei dem höchsten Peak im Spektrum nach dem Gleichanteil bei Frequenz Null gekennzeichnet. Es können auch die höherfrequenten Rauschanteile erkannt werden, die herausgefiltert werden sollen. Wählt man die Grenzfrequenz jetzt zu hoch, so werden nach der folgenden Synthese (inverse FFT) noch signifikante Rauschanteile im Signal erkennbar sein. Bei zu niedriger Grenzfrequenz werden die Maxima und Minima abgeflacht da das Signal keine genau definierte Frequenz besitzt. Es hat sich herausgestellt, daß die Wahl der Grenzfrequenz bei 2,5-facher Signalfrequenz zu den besten Ergebnissen führt. Ab dieser Grenzfrequnz werden die Frequenzanteile im Fourierspektrum auf Null gesetzt. Diese bereinigte FFT ist in Abbildung 3.6 für die Beispielmessung dargestellt.

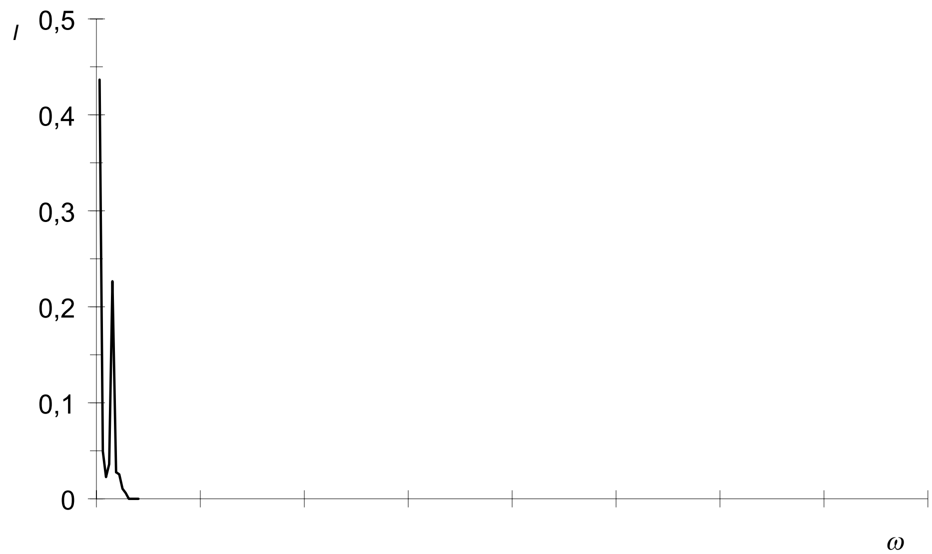

Abbildung 3.6: Bereinigte FFT der Intensitätsverteilung des Interferenzmusters

Führt man mit diesem bereinigten Spektrum eine Rücktransformation durch erhält man ein wesentlich rauschfreieres Signal (Abbildung 3.7) in dem die Extremwerte leicht ermittelt werden können.

Es ergeben sich folgende den Intensitäten entsprechende Werte

I_{max}	0,718
I_{min}	0,207

3.5.3.2 Intensitäten der Teilstrahlen

Man kann nicht von gleichen Intensitäten der beiden Teilstrahlen ausgehen. Aus diesem Grund müssen die Intensitätsverteilungen der beiden Teilstrahlen ebenfalls

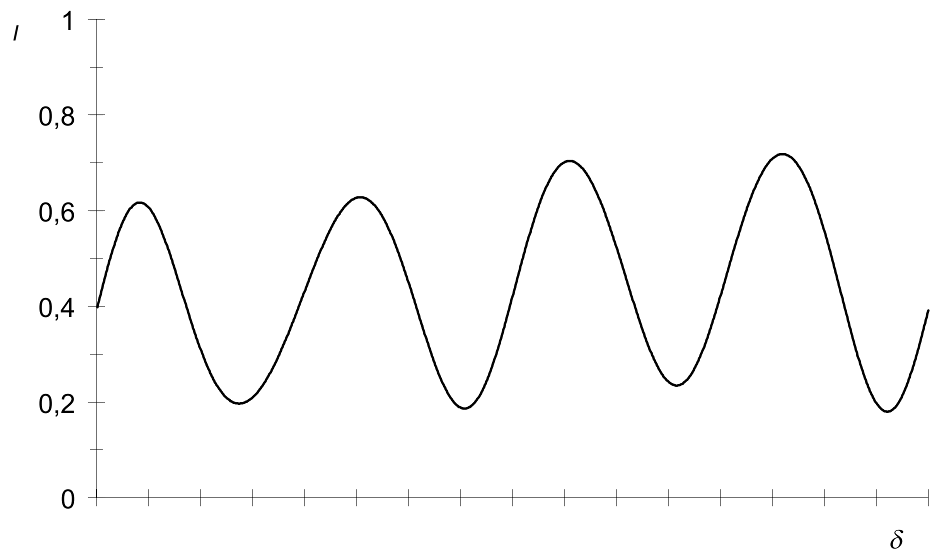

Abbildung 3.7: Gefilterte Intensitätsverteilung des Interferenzmusters

bestimmt werden. Hierzu wird wie bei der Messung eines Interferenzmusters vorgegangen, jedoch wird jeweils ein Teilstrahl abgeblendet.

Theoretisch wäre es ausreichend die Intensität eines Teilstrahls, z.B. des Referenzstrahl, zu messen. Aus dieser Verteilung und aus dem Interferenzmuster kann man die Intensität des anderen Teilstrahls berechnen. Diese Vorgehensweise hat sich aber schon zu Beginn meiner Arbeit als Fehlerquelle herausgestellt, weil Fehler durch Intensitätsschwankungen durch diese Berechnung verstärkt werden.

Die Intensitätsverteilungen der Teilstrahlen sind ebenfalls von einem Rauschen überlagert. Deswegen ist es sinnvoll, auch hier eine Filterung durch Fouriertranformation durchzuführen. In Abbildung 3.8 bis 3.11 sind die gemessenen Intensitätsverteilungen sowie die gefilterten Intensitätsverteilungen der Teilstrahlen dargestellt.

An der Stelle des Maximums im Interferenzmuster können aus den gefilterten Intesitätsverteilungen die den Intensitäten entsprechenden Werte abgelesen werden.

I_{obj}	0,326
I_{ref}	0,299

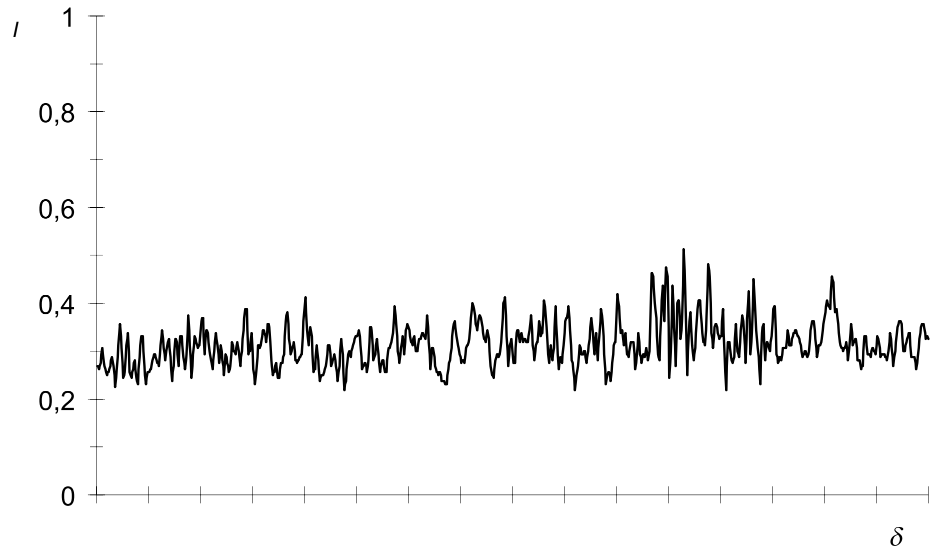

Abbildung 3.8: Gemessene Intensitätsverteilung des Objektstrahls

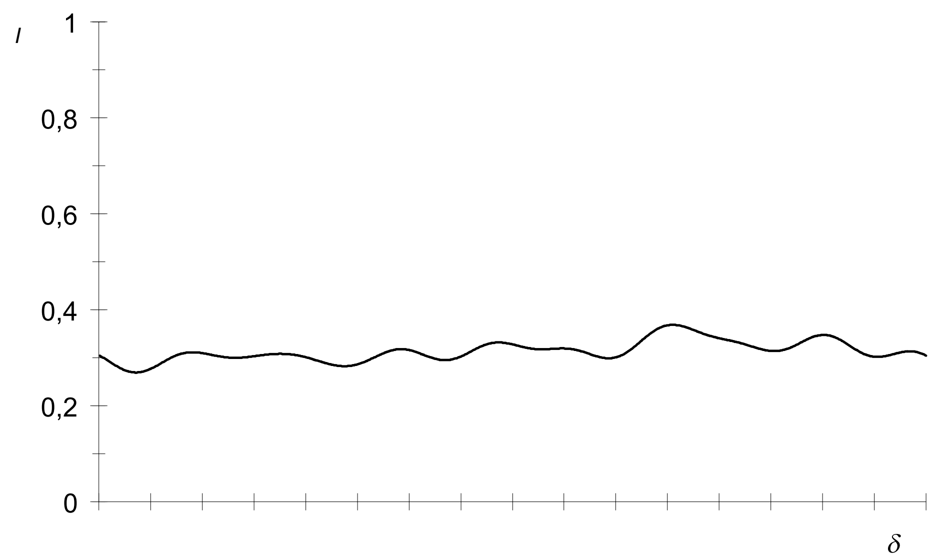

Abbildung 3.9: Gefilterte Intensitätsverteilung des Objektstrahls

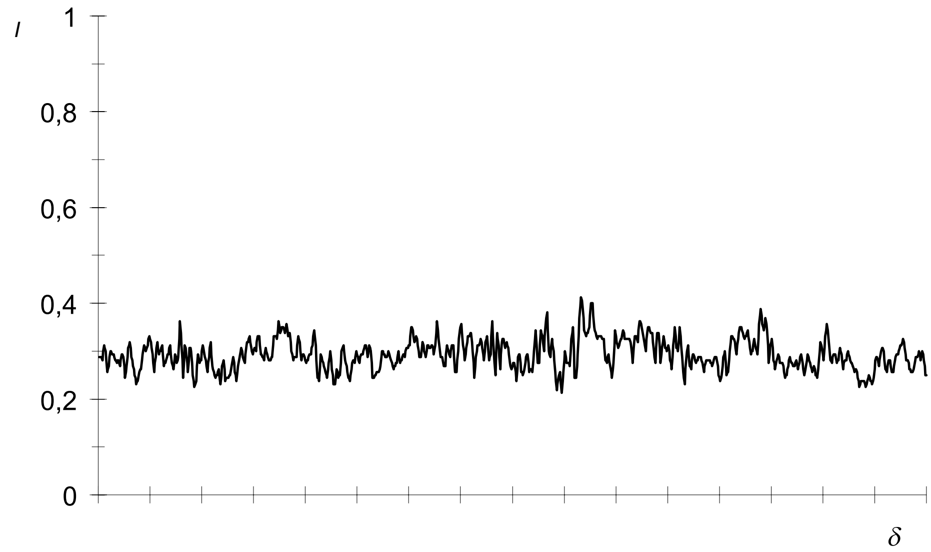

Abbildung 3.10: Gemessene Intensitätsverteilung des Referenzstrahls

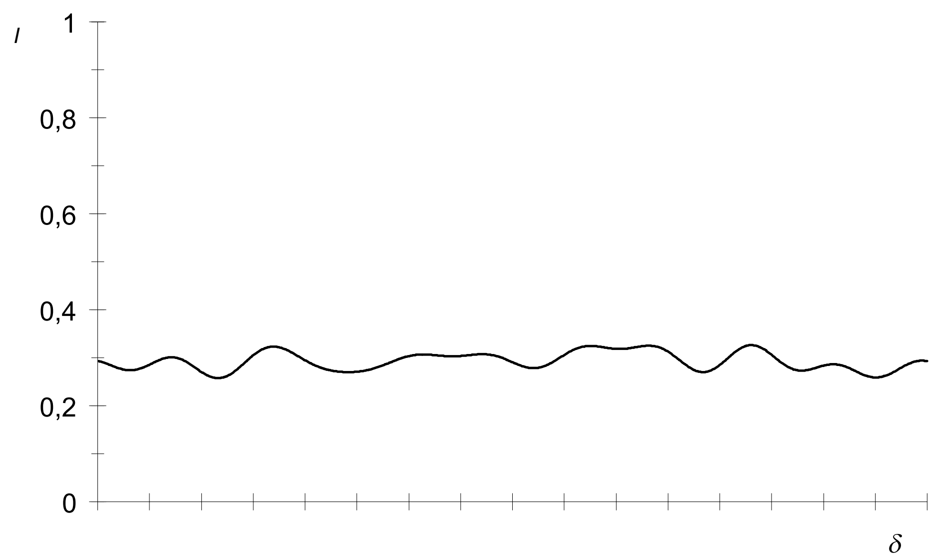

Abbildung 3.11: Gefilterte Intensitätsverteilung des Referenzstrahls

3.5.3.3 Nullintensität

Da bei der Messung der Intensitäten mit einem Oszilloskop keine Marke für die absolute Nullinie zur Verfügung steht muß eine Messung mit abgeblendeten Strahlen duchgeführt werden. Diese Nullintensität muß von den Intensitäten des Interferenzmusters und der Teilstrahlen abgezogen werden. Da sich eine relativ gleichmäßige Verteilung der Nullwerte ergibt kann man hier einen Mittelwert der Meßwerte bilden und diesen als Nullintensität definieren. Die Messung einer Nullintensität ist in Abbildung 3.12 dargestellt.

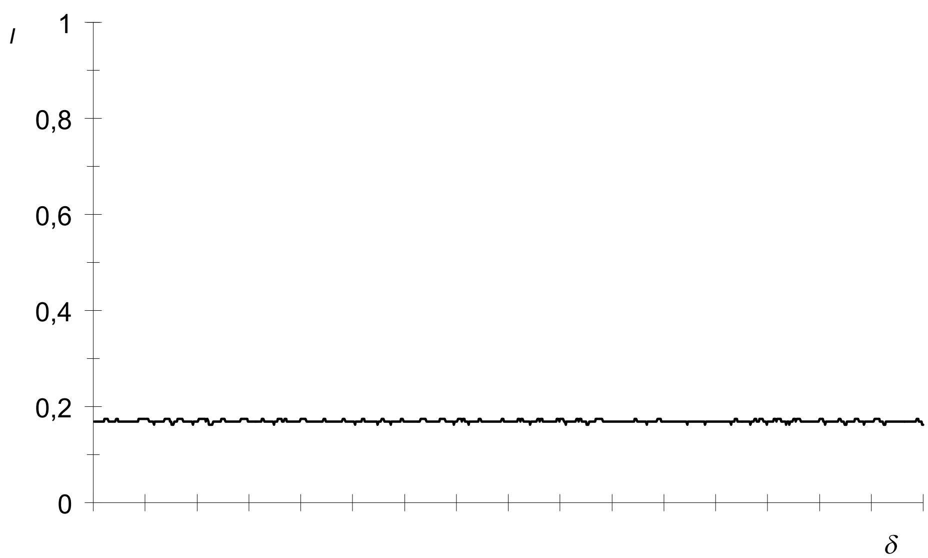

Abbildung 3.12: Messung der Nullintensität

Das der Nullintensität I_0 entsprechende Signal ergibt sich hier zu 0,172. Die den ermittelten Intensitäten entsprechenden Signale werden somit zu

I_{max}	0,546
I_{min}	0,035
I_{obj}	0,154
I_{ref}	0,127

3.5.4 Berechnung des Kohärenzgrades und der Kohärenzlänge

Aus den ermittelten Werte für I_{max} und I_{min} kann zunächst die Modulation nach (2.10) berechnet werden. Sie ergibt sich hier zu

$$M = 0,879$$

Mit I_{obj} und I_{ref} kann aus (2.26) der Kohärenzgrad für diese Wedifferenz berechnet werden.

$$\gamma = M \cdot \frac{I_{obj} + I_{ref}}{2\sqrt{I_{obj} I_{ref}}} \tag{3.2}$$

$$\gamma = 0,883$$

Dies ist das Ergebnis einer Einzelmessung bei einer Wegdifferenz von 5 cm. In der Beispielmessung wurden bei dieser Meßreihe je 10 Einzelmessungen pro Wegdifferenz durchgeführt. Aus den 10 Messungen ergibt sich für den Kohärenzgrad ein Mittelwert von

$$\gamma = 0,79 \pm 0,12$$

Hieraus kann nun die Kohärenzzeit bzw. -länge nach (2.22) berechnet werden. Da eine Berechnung aus einer Messung bei nur einer Wegdifferenz jedoch einer relativ großen Tolerenz unterliegt ist es sinnvoll, die Messung der Intensitäten bei mehreren unterschiedlichen Wegdifferenzen zu wiederholen. Somit erhält man eine Anzahl von Wertepaaren für die verschiedenen Wegdifferenzen und den entsprechenden Kohärenzgrad. An diese Werte kann jetzt die Kohärenzfunktion angeglichen werden. In diesem Beispiel wurden 9 Wegdifferenzen zwischen 0 und 25 cm vermessen. Die berechnete Kohärenzfunktion ist in Abbildung 3.13 dargestellt.

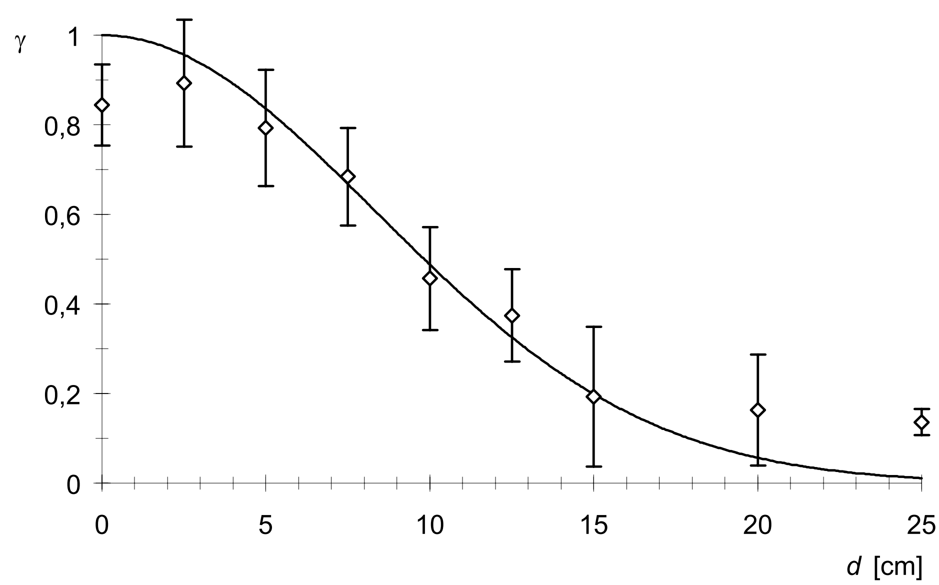

Abbildung 3.13: Kohärenfunktion Ar^{+}-Laser 514,5 nm

Für diese Meßreihe kann die Kohärenzlänge zu

$$\ell_c = 11,8 \pm 3,8 \; cm$$

berechnet werden. Auf die bei der Messung auftretenden Fehler wird in Abschnitt 4.3 eingegangen.

3.6 Meßprogramm

Das zur Kohärenzmessung erstellte Meßprogramm wurde in Borland Turbo Pascal entwickelt und besitzt eine grafische Benutzeroberfläche aus Borland Turbo Vision Standard Komponenten. Das Programm führt den Bediener im Dialog durch den Meßablauf und die anschließenden Berechnungen. Ein Benutzerhandbuch findet sich im Anhang.

Kapitel 4

Meßergebnisse

4.1 Argon-Ionen-Laser

Der vermessene Ar^+-Laser ist eine Laser der Firma Coherent, Typ Innova-90, Serien-Nr. 6041. Dieser Laser ist im Laserlabor fest installiert und wird u.a. auch für die interferometrische Längenmessung verwendet. Hierbei wird er im Monomodebetrieb genutzt. Es ist aber auch möglich durch Ausbau des Etalons und durch Verwendung einer anderen Spiegelhalterung mehrere Spektrallinien anschwingen zu lassen, man spricht dann vom Multimodebetrieb.

4.1.1 Monomodebetrieb

Im Monomodebetrieb emittiert der Ar^+-Laser Stahlung mit einer Wellenlänge von λ = 526,7 nm. Hier wurden 7 Wegdifferenzen d mit jeweils 10 Einzelmessungen vermessen.

d [cm]	γ
0	$0,94 \pm 0,06$
15	$0,99 \pm 0,03$
30	$0,94 \pm 0,15$
45	$0,91 \pm 0,09$
60	$0,92 \pm 0,17$
75	$0,95 \pm 0,09$
90	$0,94 \pm 0,16$

Tabelle 4.1: Berechnete Kohärenzgrade Ar^+-Laser Monomodebetrieb

Wie man am Meßergebenis in Abbildung 4.1 sieht liegt die Kohärenzlänge in dieser Bertriebsart außerhalb der maximal einstellbaren Wegdifferenz von 90 cm. Sie kann aus der Kohärenzfunktion zu

$$\ell_c = 218 \pm 26 \ cm$$

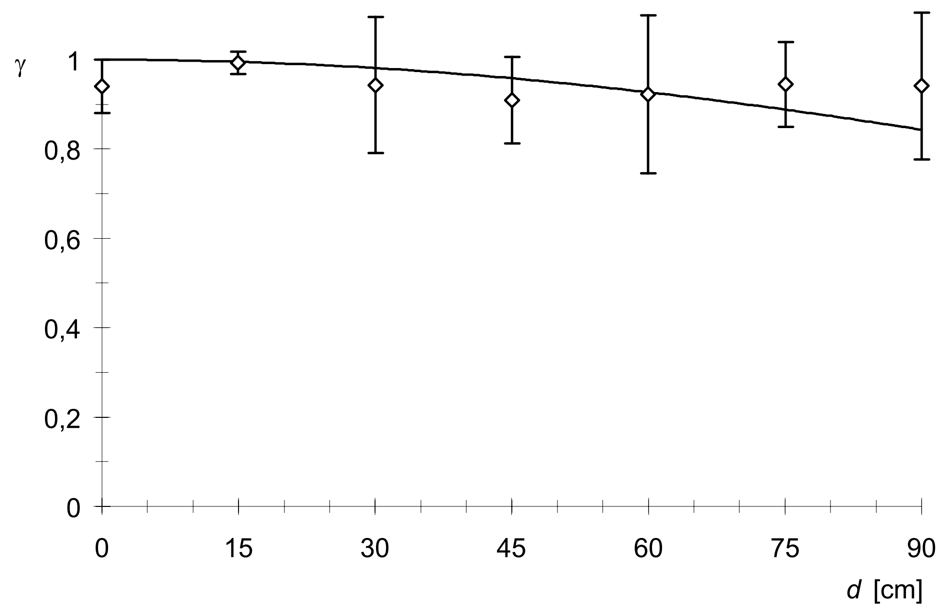

Abbildung 4.1: Kohärenfunktion Ar^+-Laser Monomodebetrieb

berechnet werden. Es kann also im Nachhinein bestätigt werden, daß der Ar^+-Laser im Monomodebetrieb für die interferometrische Längenmessung gut geeignet ist. Hier wird eine hohe Modulation des Interferenzmusters benötigt. In dieser Betriebsart kann die Modulation bei guter Justierung über die gesamte Wegdifferenz annähernd konstant gehalten werden.

Übliche Werte der Kohärenzlänge bei Ar^+-Lasern, die mit Etalon betrieben werden, liegen im Meterbereich bis hoch zu einigen zehn Metern [1]. Die Kohärenzlänge des Lasers kann in dieser Betriebsart durchaus noch über dem berechneten Wert liegen da in dem zur Berechnung zur Verfügung stehenden Bereich der Kohärenzfunktion schon kleine Änderungen der Meßwerte eine wesentlich größere Kohärenzlänge ergeben können.

Die der berechneten Kohärenzlänge entsprechende Kohärenzzeit ergibt sich nach 2.29 zu

$$\tau_c = 73 \pm 9\ ns$$

Mit 2.31 kann weiterhin die Bandbreite der emittierten Wellenlänge bestimmt werden

$$\Delta\lambda = \frac{\lambda^2}{\ell_c} \tag{4.1}$$

[1] Jeff Hecht, The Laser Guidebook [5], S. 112

$$\Delta\lambda = 0,13\ pm$$

4.1.2 Multimodebetrieb

Laut Handbuch des Lasers kann im Multimodebetrieb Strahlung in einem Wellenlängenbereich von 457,9 bis 514,5 nm emittiert werden. In diesem Bereich können sechs Hauptlinien angeregt werden [2]. Die möglichen Linien von Ar^+-Lasern in diesem Bereich sind in Tabelle 4.2 [3] angegeben.

λ [nm]
457,9
476,5
488,0
496,5
501,7
514,5

Tabelle 4.2: Ar^+-Spektrallinien (λ [nm])

Die Linien bei 488,0 nm und bei 514,5 nm bieten die mit Abstand stärksten Emissionen. Die anderen Linien können erst bei relativ hohen Ausgangsleistung von einigen Milliwatt angeregt werden. Im Folgenden sind diese beiden stärksten Linien einzeln vermessen worden, ebenso wurde eine Messung ohne Frequenzfilter durchgeführt.

4.1.2.1 Spektrallinie 488,0 nm

Mit Hilfe eines Frequenzfilters wurde in dieser Meßreihe die Spekrallinie um 488,0 nm vermessen. Hier wurden ebenfalls 10 Einzelmessungen pro Wegdifferenz durchgeführt, in dieser Meßreihe bei 16 Wegdifferenzen zwischen 0 und 15 cm. In Tabelle 4.3 sind die berechneten Werte für den Kohärenzgrad angegeben.

Die Kohärenzlänge ergibt sich hier zu

$$\ell_c = 7,7 \pm 2,5\ cm$$

Die Kohärenzlänge einer einzelnen Linie wird um 5 cm erwartet [4]. Dies kann die Messung also bestätigen.

Die Kohärenzzeit zu wird damit zu

$$\tau_c = 2,6 \pm 0,8\ ns$$

[2] Jeff Hecht, The Laser Guidebook [5], S. 104, 111

[3] Marvin J. Weber, Handbook of Laser Sience and Technology, Supplement 1: Lasers [6], S. 424ff

[4] Jeff Hecht, The Laser Guidebook [5], S. 112

d [cm]	γ
0	$0,94 \pm 0,08$
1	$0,87 \pm 0,08$
2	$0,87 \pm 0,18$
3	$0,79 \pm 0,06$
4	$0,69 \pm 0,07$
5	$0,63 \pm 0,08$
6	$0,53 \pm 0,08$
7	$0,49 \pm 0,07$
8	$0,35 \pm 0,13$
9	$0,26 \pm 0,21$
10	$0,10 \pm 0,05$
11	$0,19 \pm 0,13$
12	$0,16 \pm 0,08$
13	$0,17 \pm 0,04$
14	$0,13 \pm 0,08$
15	$0,17 \pm 0,07$

Tabelle 4.3: Berechnete Kohärenzgrade Ar^+-Laser 488,0 nm

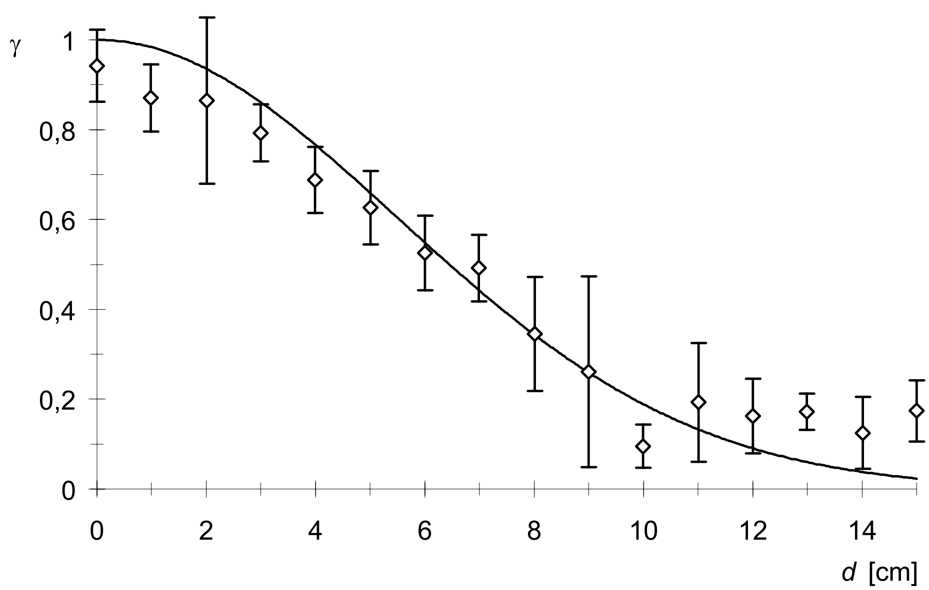

Abbildung 4.2: Kohärenfunktion Ar^+-Laser 488,0 nm

Auch hier kann Spektrale Breite in der Wellenlänge um den Mittelwert 488,0 nm angegeben werden.

$$\Delta\lambda = 3,1\ pm$$

Für die spektrale Breite einer einzelne Linie werden in der Literatur Werte um 4 pm angegeben [5].

4.1.2.2 Spektrallinie 514,5 nm

Die Messungen an der Spektrallinie um 514,5 nm sind bereits in Abschnitt 3.5 beschrieben. Hier sollen nur kurz die Ergebnisse angeführt werden. Eine Abbildung der Kohärenzfunktion findet sich auf Seite 27.

d [cm]	γ
0,0	$0,84 \pm 0,09$
2,5	$0,89 \pm 0,14$
5,0	$0,79 \pm 0,13$
7,5	$0,68 \pm 0,11$
10,0	$0,46 \pm 0,11$
12,5	$0,37 \pm 0,10$
15,0	$0,19 \pm 0,15$
20,0	$0,16 \pm 0,12$
25,0	$0,13 \pm 0,03$

Tabelle 4.4: Berechnete Kohärenzgrade Ar^+-Laser 514,5 nm

$$\ell_c = 11,8 \pm 3,8\ cm$$

$$\tau_c = 3.9 \pm 1,3\ ns$$

$$\Delta\lambda = 2,2\ pm$$

Diese Werte sind erwartungsgemäß ähnlich wie bei der Spektrallinie um 488,0 nm.

4.1.2.3 Beide Linien

In dieser Meßreihe wurde im Multimodebetreib kein Frequenzfilter eingesetzt, es wurden die beiden angeregten Linien gemeinsam vermessen.

$$\ell_c = 6,1 \pm 2,2\ cm$$

[5] Jeff Hecht, The Laser Guidebook [5], S. 111

d [cm]	γ
0	$0,51 \pm 0,18$
2	$0,86 \pm 0,13$
4	$0,56 \pm 0,18$
6	$0,45 \pm 0,11$
8	$0,31 \pm 0,10$
10	$0,19 \pm 0,09$
12	$0,15 \pm 0,09$
14	$0,18 \pm 0,07$

Tabelle 4.5: Berechnete Kohärenzgrade Ar^+-Laser beide Linien

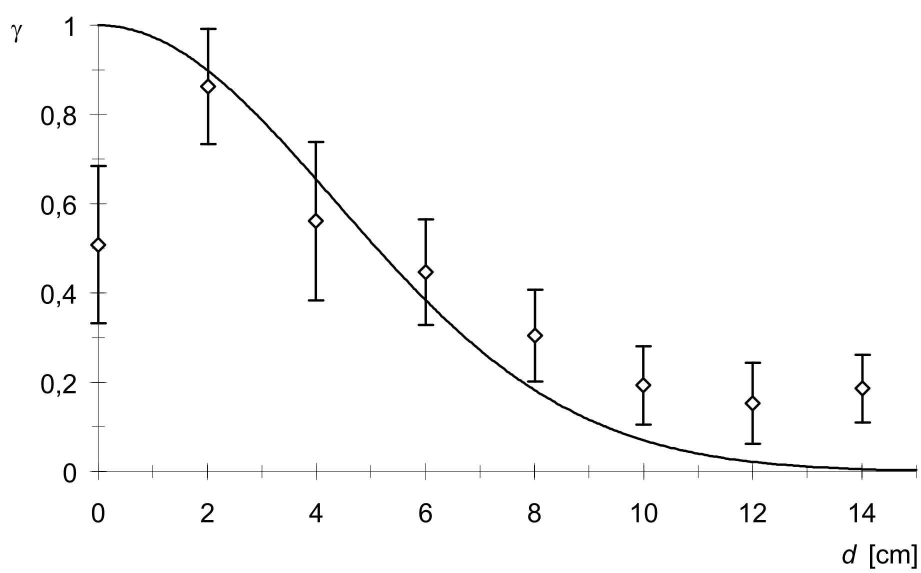

Abbildung 4.3: Kohärenfunktion Ar^+-Laser beide Linien

$$\tau_c = 2,0 \pm 0,7 \; ns$$

In diesem Fall ist eine Angabe der spektralen Breite nicht sinnvoll.

4.2 Helium-Neon Laser

HeNe-Laser emittieren Ihre Strahlung bei einer Wellenlänge von 632,8 nm. Hier wurde eine Meßreihe an einem HeNe-Laser der Firma Melles Griot vom Typ 05-LHP-911, Serien-Nr. 3552AR durchgeführt. Dieser Laser wird im Laserlabor an einer Laser-Doppler-Aneometrie Meßanlage verwendet.

d [cm]	γ	*d* [cm]	γ
0	0,98	22	0,45
2	1,13	24	0,28
4	1,06	26	0,35
6	1,06	28	0,28
8	0,93	30	0,31
10	0,83	35	0,21
12	0,79	40	0,19
14	0,75	45	0,26
16	0,69	50	0,25
18	0,66	55	0,18
20	0,39	60	0,24

Tabelle 4.6: Berechnete Kohärenzgrade HeNe-Laser

Die Kohärenzlänge berechnet sich für den HeNe-Laser zu

$$\ell_c = 29,1 \; cm$$

Handelsübliche HeNe-Laser haben Kohärenzlängen um 20 bis 30 cm [6]. Andere Angabe gehen von größer 30 cm aus [7]. Diese Werte können ebenfalls bestätigt werden. Die Kohärenzzeit beträgt hier

$$\tau_c = 9,7 \; ns$$

und für die spektrale Breite ergibt sich

$$\Delta\lambda = 1,4 \; pm$$

[6] Jeff Hecht, The Laser Guidebook [5], S. 93

[7] Holger Venzke, Kohärenz-Rader: Ein aperturunabhngiges Verfahren zur 3D-Formerfassung optisch rauher Objekte [11], S. 23

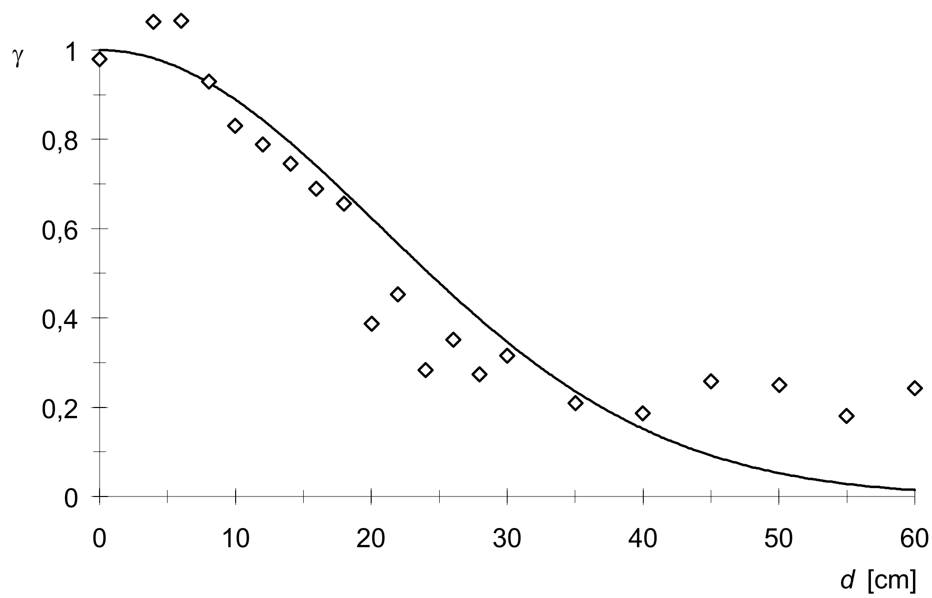

Abbildung 4.4: Kohärenfunktion HeNe-Laser

Dieser Wert deckt sich relativ gut mit Literaturangaben, die von einer Breite der Spektrallinie von ca. 1,9 pm ausgehen [8].

4.3 Fehlerbetrachtung

Die angegebenen Meßergebnisse unterliegen alle einer großen Toleranz. In diesem Abschnitt soll nun auf die möglichen Fehlerquellen eingegangen werden. Den Meßprozeß betreffend können folgende Punkte angeführt werden:

- Bei der Einstellung der Wegdifferenz muß man eine Unsicherheit von ca. 1 mm in Betracht ziehen. Diese Ungenauigkeit würde den berechneten Kohärenzgrad einer Strahlungsquelle mit einer Kohärenzlänge von 5 cm um maximal 1 % verfälschen. Für größere Kohärenzlängen wird dieser Fehler kleiner. Bei einer Kohärenzlänge von 20 cm wären es nur noch 0,5 %. Die Wegdifferenzen über den Strahlquerschnitt im μm-Bereich können hier vernachlässigt werden.

- Durch die Digitalisierung des Meßsignals im Oszilloskop wird die Signalhöhe der vertikalen Auflösung des Oszilloskops entsprechend in 256 diskrete Werte unterteilt. Bei Ausnutzung der halben vertikalen Auflösung für das Meßsignal bleibt der Digitalisierungsfehler noch unter 1 %. Dieser Fehler setzt sich

[8] Jeff Hecht, The Laser Guidebook [5], S. 92

in der weiteren Berechnung fort und wird den Kohärenzgrad um maximal 2,5 % verfälschen.

- Bei der Filterung durch FFT und inverse FFT kann durch die Wahl der Grenzfrequenz die Höhe der Extremwerte verändert werden. In einem Bereich der Grenzfrequenz von 2- bis 3-facher Signalfrequenz verändern sich die Extremwerte um ca. 1 %. Eine andere Wahl der Grenzfrequenz scheidet aus, da sonst die Extremwerte entweder extrem abgeflacht werden oder das Rauschen deutliche Auswirkungen zeigt. Durch die weitere Berechnung ergibt sich also wie bei dem Digitalisierungfehler eine Unsicherheit von 2,5 %.

Insgesamt können die Fehler durch die Meßanlage etwa 5 % betragen. Einen wesentlich größeren Einfluß auf das Ergebnis haben allerdings die Eigenschaften der zu vermessenden Strahlung selbst. In Abbildung 4.5 sind die gefilterten Signale von drei Interferenzmessungen am Ar^+-Laser dargestellt. Die Messungen wurden im Abstand von je einer Minute aufgenommen. Hierbei wurde an der Justierung keine Änderung vorgenommen.

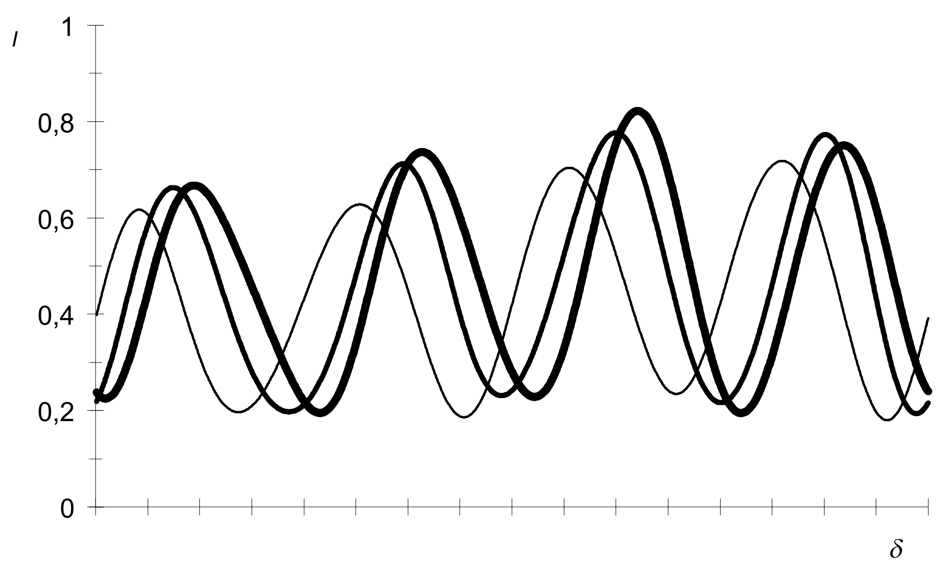

Abbildung 4.5: Vergleich von 3 Interferenzmessungen

Zunächst läßt sich hier eine starke Änderung der Gesamtintensität feststellen. Diese Intensitätsschwankungen sind in der Anlaufphase des Lasers noch wesentlich höher. Wichtiger für die Kohärenzmessung ist jedoch eine erkennbare Veränderung der Modulation des Streifenmusters. Das Verhältnis der Intensitäten der beiden Teilstrahlen wird sich nicht verändern, also ist eine Änderung der Modulation auf eine Änderung des Kohärenzgrades zurückzuführen. Dies bedeutet

wiederum, daß die Kohärenzlänge bzw. -zeit nicht konstant bleibt. Hierauf sind auch die großen Toleranzen in den Ergebnissen zurückzuführen.

Will man die statistischen Schwankungen der Strahlungsquelle im Ergebnis minimieren, dann sind sehr viele Messungen über einen längeren Zeitraum nötig.

Kapitel 5

Schlußbemerkungen

5.1 Verbesserungsvorschläge

Will man Laser mit einer Kohärenzlänge über 90 cm vermessen, so könnte zur Ermittlung von Werten für γ bei Wegdifferenzen im Bereich der Kohärenzlänge und darüber eine Verlängerung der optischen Wegdifferenz hinzugefügt werden. Hierzu könnte beispielsweise durch ein Prisma der Strahl aus der Strahlführung des Objektstrahls ausgekoppelt werden. Mit einem weiteren Prisma könnte er dann zurückgeworfen und durch das erste Prisma wieder in den ursprünglichen Strahlengang eingekoppelt werden. Die mechanische Baulänge dieser Komponenten würde als vierfache optische Wegdifferenz eingehen, da sie zweimal vom Objekstrahl durchlaufen wird. Diese Komponenten würde jedoch eine Justierung des Aufbaus erschweren.

Zur Erhöhung des Meßkomforts wäre es wünschenswert, die Digitalisierung des Signals von einen A/D-Wandler durchfüren zu lassen. Dadurch bestünde die Möglichkeit, die Auslesung der CCD-Kamera direkt vom PC zu steuern. Dies könnte die Meßzeiten wesentlich verkürzen. Weiterhin entfielen Fehlmessungen, die durch unzureichende Einstellungen am Oszilloskops entstehen können. Im Meßprogramm müßte hierzu nur die Unit GetData von einer Auslesung der seriellen Schnitstelle in eine Auslesung des A/D-Wndlers umgeschrieben werden. Weiterhin müßten die Konstanten, die die vertikale und horizontale Auflösung festlegen, an die Werte der CCD-Kamera angepaßt werde. Alle anderen Komponenten sind so geschrieben, daß sie weiterhin arbeiten. Da die CCD-Kamera nicht fest in den Aufbau eingebaut ist könnte sie auch für andere Meßzwecke herangezogen werden.

Die Abblendmöglichkeiten der Teilstrahlen zum vermessen der Intensitätsverteilung nur eines Strahls könnten ebenfalls als PC-gesteuerte Blenden installiert werden. Dadurch könnte eine versehenliche Dejustierung bei der Abblendung der Strahlen verhindert werden.

Im Meßprogramm könnten noch einige Details ergänzt werden. Zum Beispiel wäre das Anlegen eines Verzeichnisses für eine Meßreihe aus der Oberfläche des Programmes heraus wünschenswert. Ebenso könnte das Ändern einer Protokoll-

datei durch einen integrierte Editor erleichtert werden (vgl. Benutzerhandbuch im Anhang).

Das Programm zur interferometrischen Längenmessung könnte als Menüpunkt in das Meßprogramm aufgenommen werden. Somit bestünde eine einheitliche Bedienung beider Meßverfahren an diesem Aufbau.

5.2 Zusammenfassung

Das über eine einstellbare Wegdifferenz an dem Michelson-Interferometer erzeugte Interferenzstreifenmuster kann durch eine CCD-Karera detektiert werden. Aus der Modulation des Streifenmusters und aus den Intensitäten der Teilstrahlen wird die Kohärenzfunktion berechnet. Entspricht die Wegdifferenz der Kohärenzlänge, so wird der Kohärenzgrad zu $1/e$.

Mit dem Interferometer-Aufbau, der von mir weitgehende übernommen wurde, können problemlos Interferenzmuster erzeugt werden. Mit der CCD-Kamera können Interferenzmuster an einer stationären Position detektiert werde. Die Meßwertübertragung über das Oszilloskop ist zwar umständlich, bietet aber auch Vorteile. Das Meßprogramm arbeitet zuverlässig. Bei den vermessenen Lasern konnten die zu erwartenden Ergebnisse erreicht werden.

Die Anlage ist auf dem jetzigen Stand im Praktikum an der Fachhochschule einsetzbar, jedoch könnten noch einige Verbesserungen vorgenommen werden.

Es wäre weiterhin interessant, die Kohärenzeigenschaften von Laserdioden zu untersuchen. Dies könnte eine geeignete Aufgabe im Rahmen eines Praktikums sein.

Anhang A

Zeichnungen

Anhang B

Datenblatt der CCD-Kamera

Anhang C

Benutzerhandbuch Meßprogramm

Das Meßprogramm wurde als Turbo Vision Standard Objekt definiert, somit steht dem Benutzer eine für Turbo Vision übliche Umgebung zur Verfügung. Die Bedienung kann mit der Maus oder über die Tastatur erfolgen. Beim Start erscheint eine kurze Programminformation.

In der Menüleiste sind die Programmoptionen `Messung`, `Konvertieren`, `Anzeige`, `Berechnung`, `Motorsteuerung` und `Etc` (Programminformation und Taschenrechner) sowie eine Uhr untergebracht. Die Statuszeile enthält Informationen über Tastaturbefehle.

Über `F10` läßt sich die Menüleiste aktivieren. Die Programmoptionen können ebenfalls mit der Tastenkombination `Alt + <Markierter Buchstabe>` aufgerufen werden. Mit der `Esc`-Taste lassen sich Fenster schließen.

Beendet wird das Programm mit der Tastenkombination `Alt + X` oder mit der Maus durch anklicken von `Exit`.

C.1 Aufnehmen einer neuen Meßreihe

Soll eine neue Meßreihe an einer Strahlungsquelle aufgenommen werden ist es zunächst sinnvoll, für diese Meßreihe ein eigenes Verzeichnis anzulegen da i.a. relativ viele Dateien erstellt werden. Über den Menüpunkt `Messung` wird zunächt ein Dialogfenster geöffnet, in dem der Name für die Protokolldatei dieser Meßreihe vergeben wird. Hier kann ein Name gewählt werden, der die Strahlungsquelle beschreibt. Die Erweiterung `DAT` darf nicht verändert werden. Diese Protokolldatei muß sich im Meßverzeichnis befinden. In der Protokolldatei werden die Namen der Interferenz-, Objektstrahl-, Referenzstrahl- und Nullintensitätsmessung für jede Wegdifferenz sowie die Anzahl der Einzelmessungen gespeichert.

Wurde die Protokolldatei korrekt bezeichnet wird der Benutzer im nächsten Dialogfenster aufgefordert, die Dateinamen für Interferenz-, Objektstrahl-,

Referenzstrahl- und Nullintensitätsmessung (ohne Erweiterung) einzugeben. Bei der ersten Messung sind in den Eingabefeldern als Standardnamen `i0`, `o0`, `r0` und `d0` eingeblendet. Diese können ensprechend geändert werden. In einem weiteren Eingabefeld ist die Wegdifferenz (in cm) einzugeben. Die Anzahl der Einzelmessungen wird ebenfalls abgefragt. Dieser Wert kann über eine gesamte Meßreihe nicht verändert werden, also muß bei der ersten Messung entschieden werden. Wird mehr als eine Einzelmessung gewünscht dürfen die Dateinamen der Messungen keine 8 Zeichen mehr lang sein da jeweils eine entsprechende Nummer angehängt wird. Wenn alle Angaben korrekt sind startet der Meßvorgang, ansonsten erscheint eine Fehlermeldung und die Eingaben können entsprechend geändert werden.

Im folgenden Fenster wird der Benutzer aufgefordert, die `PLOT`-Taste am Oszilloskop zum einlesen der Messdaten zu drücken. Es ist jeweils die Nummer und die Art der Messung angegeben, so daß gegebenenfalls Teilstrahlen ausgeblendet werden müssen. Hier werden alle Messungen für eine Wegdifferenz nacheinander abgearbeitet. Der Meßvorgang kann jederzeit mit der `ESC`-Taste abgebrochen werden.

Nach Beendigung der Messung besteht die Option eine weitere Messung aufzunehmen oder abzubrechen. Bei einer weitere Messung müssen die Dateinamen und die Wegdifferenz in den Eingabefelder entsprechend geändert werden (Einstellung einer neuen Wegdifferenz siehe C.6). Sollen die Objektstrahl-, die Referenzstrahl- oder die Nullintensitätsmessung nicht erneut aufgenommen werden können Ihre Dateinamen unverändert bleiben. Sie werden dann nach einer Abfrage in die Protokolldatei aufgenommen. Dies kann je nach Art der Strahlungsquelle und Veränderungen in der Justierung entschieden werden. Der Meßvorgang startet wieder nach Prüfung auf Korrektheit aller Eingaben.

Dieser Meßdialog wird solange wiederholt bis der Benutzer entscheidet, keine weitere Messung aufzunehmen.

C.2 Hinzufügen von Messungen zu einer Meßreihe

Soll eine bereits bestehende Meßreihe um neue Messungen erweitert werden, so ist über den Menüpunkt `Messung` das Dialogfenster zur Auswahl einer Protokolldatei aufzurufen. Hier kann jetzt eine bestehende Protokolldatei geöffnet werden. Es wird ein Abfrage eingeblendet, ob neue Messungen hinzugefügt werden sollen. Wird dies bejaht, so öffnet sich das Dialogfenster zur Eingabe der Dateinamen der Einzelmessungen und der Wegdifferenz. Hier sind als Vorgaben die Daten der letzten Messung dieser Meßreihe eingeblendet. Nach entsprechender Änderung dieser Daten erfolgt der Meßvorgang analog zur Aufnahme einer neuen Meßreihe (siehe C.1).

Wurde bereits die Funktion des Kohärenzgrades berechnet, dann ist die Ergebnisdatei zu löschen (siehe C.7).

C.3 Konvertieren einer Plot-Datei

Die Messdaten liegen nach der Messung in Form einer Plotdatei vor. Über die Menüoption `Konvertieren` kann die Plotdatei einer Einzelmessung in einem Dialogfenster ausgewählt werden, die Plotdaten werden dann in Integerwerte konvertiert und in einer Datei vom Typ `MES` abgelegt. Diese Konvertierung muß jedoch nicht explizit ausgeführt werden da sie bei der Anzeige einer Einzelmessung oder bei der Berechnung der Kohärenzlänge aus einer Meßreihe automatisch im Hintergrund durchgeführt wird.

C.4 Anzeigen einer Messung

Durch anwählen der Menuoption `Messung` wird ein Dialogfenster geöffnet, in dem als Standardvorgabe eine Plotdatei geöffnet werden kann. Wird eine Datei ausgewählt, so erscheint ein Auswahlfenster, in dem die Anzeigeoption gewählt werden kann. Es können neben den Originalmessdaten einer Einzelmessung ihre Fouriertransformation sowie die gefilterte Intensitätsverteilung ausgewählt werden. Nach dem Auswählen einer Option werden die Daten grafisch dargestellt. Der Grafikbildschirm wird mit der `Esc`-Taste wieder geschlossen. Wenn die Fouriertranformation oder die gefilterte Intensitätsverteilung dargestellt wurde erscheint jetzt ein Anfragefenster, ob die berechneten Daten gespeichert werden sollen. Die Daten werden bei Zustimmung in einer Datei vom Typ `FFT` bzw. `KOR` abgelegt. Eine Speicherung erspart bei wiederholter Darstellung derselben Messung eine erneute Berechnung.

Neben Dateien vom Typ `PLT` können ebenfalls Dateien von Typ `MES`, `FFT` und `KOR` geöffnet werden sofern diese bereits erstellt wurden.

C.5 Anzeigen der Kohärenzfunktion einer Meßreihe

Soll eine Darstellung des Kohärenzgrades in Abhängigkeit von der Wegdifferenz dargestellt werden, so ist die Menüoption `Berechnung` zu wählen. Hier kann in einem Dialogfenster die Protokolldatei der gewünschten Meßreihe ausgewählt werden. Falls die Ergebnisdatei (siehe C.7) noch nicht berechnet wurde geschieht dies jetzt. In einem Informationsfenster wird der Fortschritt der Berechnung angezeigt. Bei einer umfangreichen Meßreihe kann die Berechnung durchaus einige Minuten dauern. Die Berechnung kann mit der `Esc`-Taste abgebrochen werden.

In der Grafikausgabe werden der Kohärenzgrad mit Standardabweichung für die einzelnen Meßpunkte sowie eine durch diese Werte berechnete Funktion dargestellt. In dieser Funktion ergibt sich beim Wert $1/e$ die Kohärenzlänge, die ebenfalls mit Standardabweichung in der rechten oberen Ecke angegeben ist. Das Schließen des Grafikbildschirms erfolgt wiederum mit der `Esc`-Taste.

C.6 Motorsteuerung

Eine zu vermessende Wegdifferenz kann über den Menüpunkt `Motorsteuerung` eingestellt werden. Über die Tasten `v` und `r` kann der Wagen vorwärts und rückwärts bewegt werden. Der Wagen wird durch drücken einer beliebigen Taste wieder gestoppt. Mit den Cursortasten <- und -> kann der Wagen in Einzelschritten in beide Richtungen bewegt werden. Das Fenster zur Motorsteuerung wird mit der `Esc`-Taste wieder geschlossen.

C.7 Dateitypen

Im Meßprogramm können sechs Dateitypen auftreten, ihre Bedeutung ist im Folgenden beschrieben.

PLT Da vom Oszilloskop ein Plotsignal übertragen wird, werden die Messdaten zunächst in einem HPGL-kompatiblen Format `PLT` abgelegt. Diese Dateien können auch direkt aus einem digitalen Plotter vom Typ HP7550 ausgegeben werden.

MES Nach der Konvertierung einer Plot-Datei erhält man eine Datei vom Typ `MES`, in der die Messdaten als Integerwerte gespeichert sind.

FFT Dateien Vom Typ `FFT` enthalten die Daten der Fouriertransformation einer Einzelmessung. Die Werte sind vom Typ Real.

KOR Die Daten einer gefilterten Intensitätsverteilung können in einer Datei vom Typ `KOR` ebenfalls als Realwerte gespeichert werden.

DAT In der Protokolldatei (`DAT`) einer Meßreihe werden die Informationen über die zu der Meßreihe gehörenden Einzelmessungen gespeichert. In den ersten zwei Zeilen stehen die Anzahl der Wegdifferenzen, für die Messungen aufgenommen wurden, und die Anzahl der Einzelmessungen pro Wegdifferenz. Hierauf folgen für jede Wegdifferenz ein Datensatz, der den Namen der Nullintensitäts-, der Referenzstrahl-, der Objektstrahl- und der Interferenzmessung sowie die Wegdifferenz in cm enthält. Diese Datei kann nötigenfalls per Hnad editiert werde, z.B. wenn eine Messung aus einer Meßreihe gelöscht werden soll. Hierbei ist unbedingt darauf zu achten, daß die Anzahl der Wegdifferenzen mit der Anzahl der Datensätze übereinstimmt und daß je eine Leerzeile zwischen den Datensätzen erhalten bleibt. Sofern eine Ergebnisdatei (s.u.) vorhanden ist sollte diese gelöscht und neu berechnet werden.

GAM Das Ergebnis der Berechnung einer Meßreihe wird in einer Datei vom Typ `GAM` gespeichert. Diese Datei hat den gleichen Namen wie die Protokolldatei. In dieser Datei stehen am Anfang die berechnete Kohärenzlänge und

die Standardabweichung der Kohärenzlänge. Danach werden für jeden Abstand die Wegdifferenz, der berechnete Kohärenzgrad und die Standardabweichung des Kohärenzgrades aufgeführt. Wurde nur je eine Messung pro Wegdifferenz durchgeführt haben die Standardabweichungen den Wert 0.

Anhang D

Programmlisting

D.1 Programm CM

```
{-------------------------------------------------------------------}

{ Programm zur Messung der Kohaerenzlaenge von Laserstrahlung
  Diplomarbeit
  Fachhochschule Wedel
  Udo Becker
  26. August 1997 }
program CM;

{$X+}
{$S+}
{$M 65520,8192,655360}

{-------------------------------------------------------------------}

uses
  Anzeige,     { Grafische Ausgabe }
  ConvertU,    { Konvertierung von Plot-Daten }
  Etc,         { zusaetzliche Programmteile }
  Glob,        { Globale Deklarationen }
  MessU,       { Einlesen der Messdaten }
  MotorU,      { Motorsteuerung }

  { Turbo Pascal Standard Units }
  App, Drivers, Gadgets, Memory, Menus, Objects, Views;

type
  TCM = object(TApplication)
```

```
    Clock : PClockView;
    Heap  : PHeapView;
    constructor Init;
    procedure HandleEvent(var Event : TEvent); virtual;
    procedure Idle; virtual;
    procedure InitMenuBar; virtual;
    procedure InitStatusLine; virtual;
  end;

const
  HeapSize = 48 * (1024 div 16); { 48k Heap }

{---------------------------------------------------------------------}

{ Initialisierung }
constructor TCM.Init;

var
  R : TRect;

begin
  MaxHeapSize := HeapSize;
  inherited Init;
  GetExtent(R);
  R.A.X := R.B.X - 9;
  R.B.Y := R.A.Y + 1;
  Clock := New(PClockView, Init(R));
  Insert(Clock);
  GetExtent(R);
  Dec(R.B.X);
  R.A.X := R.B.X - 9;
  R.A.Y := R.B.Y - 1;
  Heap := New(PHeapView, Init(R));
  Insert(Heap);
  Info;
end; { Init }

{---------------------------------------------------------------------}

{ Auswahl der Handlungsmoeglichkeiten }
procedure TCM.HandleEvent(var Event: TEvent);

begin
  inherited HandleEvent(Event);
```

```
  case Event.What of evCommand:
    begin
      case Event.Command of
        cmMessung : Messdialog;
        cmConvert : ConvertFile;
        cmAusgabe : Grafikausgabe;
        cmGamma   : LCDarstellung;
        cmMotor   : Motorsteuerung;
        cmRechner : Rechner;
        cmInfo    : Info;
      else
        Exit;
      end;
      ClearEvent(Event);
    end;
  end;
end; { HandleEvent }

{---------------------------------------------------------------------}

{ Uhrzeit und Speicherbelegung aktualisieren }
procedure TCM.Idle;

begin
  inherited Idle;
  Clock^.Update;
  Heap^.Update;
end; { Idle }

{---------------------------------------------------------------------}

{ Menueleiste }
procedure TCM.InitMenuBar;

var
  R : TRect;

begin
  GetExtent(R);
  R.B.Y := R.A.Y + 1;
  MenuBar := New(PMenuBar, Init(R, NewMenu(
    NewItem('~M~essung', '', kbNoKey, cmMessung, hcMessung,
    NewItem('~K~onvertieren', '', kbNoKey, cmConvert, hcConvert,
    NewItem('~A~nzeige', '', kbNoKey, cmAusgabe, hcAusgabe,
```

```
    NewItem('~B~erechnung', '', kbNoKey, cmGamma, hcGamma,
    NewItem('Motor~s~teuerung', '', kbNoKey, cmMotor, hcMotor,
    NewSubMenu('~E~tc', hcNoContext, NewMenu(
      NewItem('~T~-Rechner', '', kbNoKey, cmRechner, hcRechner,
      NewItem('~I~nfo', '', kbNoKey, cmInfo, hcInfo,
      nil))),
    nil)))))))));
end; { InitMenuBar }

{------------------------------------------------------------------}

{ Statuszeile }
procedure TCM.InitStatusLine;

var
  R : TRect;

begin
  GetExtent(R);
  R.A.Y := R.B.Y - 1;
  StatusLine := New(PStatusLine, Init(R,
    NewStatusDef(0, $FFFF,
      NewStatusKey('~Alt-X~ Exit', kbAltX, cmQuit,
      NewStatusKey('~F10~ Menu', kbF10, cmMenu,
      NewStatusKey('~Esc~ Fenster schlieen', kbEsc, cmClose,
      nil))),
    nil)));
end; { InitStatusLine }

{------------------------------------------------------------------}

var
  ProgCM : TCM;

begin { CM }
  New(Imes);
  New(Itrans);
  New(IRe);
  New(IIm);
  with ProgCM do
  begin
    Init;
    Run;
    Done;
```

```
  end;
  Dispose(Imes);
  Dispose(Itrans);
  Dispose(IRe);
  Dispose(IIm);
end. { CM }

{----------------------------------------------------------------------}
```

D.2 Unit Glob

```
{----------------------------------------------------------------------}

{ Deklaration globaler Konstanten, Typen, Variablen, Prozzeduren und
  Funktionen }
unit Glob;

{$D-}

{----------------------------------------------------------------------}

interface

const
  cmInfo    = 1001;
  cmRechner = 1002;
  cmMessung = 1003;
  cmConvert = 1004;
  cmAusgabe = 1005;
  cmGamma   = 1006;
  cmMotor   = 1007;
  hcRechner = 1;
  hcInfo    = 2;
  hcMessung = 3;
  hcConvert = 4;
  hcAusgabe = 5;
  hcGamma   = 6;
  hcMotor   = 7;

  ExtPlot  = 'PLT';
  ExtMess  = 'MES';
  ExtFFT   = 'FFT';
  ExtKor   = 'KOR';
```

```
  ExtDaten = 'DAT';
  ExtGamma = 'GAM';

  nmax     = 1024;
  hmax     = 256;

type
  RealArray = array [0..nmax - 1] of single;
  Wordarray = array [0..nmax - 1] of word;
  DatFile = record
    int, obj, ref, dark : string;
  end;
  Werte = record
    gamma, SAM, delta : real;
  end;
  RMessdaten = record
    files : array [0..3] of string [8];
    dates : array [0..1] of string [8];
  end;

var
  df                                 : array [0..30] of DatFile;
  w                                  : array [0..30] of Werte;
  Messdaten                          : RMessdaten;
  Imes                               : ^Wordarray;
  Itrans, IRe, IIm                   : ^Realarray;
  Anzahldelta, FFTmax, AnzeigeSchalter : word;
  AnzahlMessung, Fehler              : integer;
  deltac, deltacSAm, Tau             : real;
  FileName, Path, Extens             : string;
  Abbruch, Ok                        : boolean;

function Exists(Fn : string) : boolean;
procedure ValueR(Param : string; var a : real);
procedure ValueI(Param : string; var i : integer);

{---------------------------------------------------------------------}

implementation

{---------------------------------------------------------------------}

{ Pruefung ob eine Datei vorhanden ist }
function Exists(Fn : string) : boolean;
```

```
var
  Datei : file;

begin
  Assign(Datei, Fn);
  {$I-}
  Reset(Datei);
  {$I+}
  if IOResult = 0 then
  begin
    Exists := true;
    Close(Datei);
  end
  else Exists := false;
end; { Exists }

{--------------------------------------------------------------------}

{ Konvertierung eines String in einen Real-Wert }
procedure ValueR(Param : string; var a : real);

begin
  Val(Param, a, Fehler);
  Ok := Ok and (Fehler = 0)
end; { ValueR }

{--------------------------------------------------------------------}

{ Konvertierung eines String in einen Integer-Wert }
procedure ValueI(Param : string; var i : integer);

begin
  Val(Param, i, Fehler);
  Ok := Ok and (Fehler = 0)
end;

{--------------------------------------------------------------------}

end. { Glob }

{--------------------------------------------------------------------}
```

D.3 Unit MessU

```
{------------------------------------------------------------------}

{ Messdialog, Auswahl einer Protokolldatei, Einlesen der Messdaten }
unit MessU;

{$I-}

{------------------------------------------------------------------}

interface

procedure Messdialog;

{------------------------------------------------------------------}

implementation

uses
  Async4U1,    { Ansteuerung der seriellen Schnittstelle }
  Glob,        { Globale Deklarationen }
  LoadSave,    { Laden und Speichern von Dateien }

  { Turbo Pascal Standard Units }
  App, Crt, Dialogs, MsgBox, Objects, StdDlg, Views;

var
  mes, s : string;

{------------------------------------------------------------------}

{ Einlesen der Messdaten }
procedure ReadOut;

const
  timeout = 4;

var
  R        : TRect;
  DD       : PDialog;
  s1       : string;
  c, Taste : char;
  received : boolean;
```

```
  f         : Text;

begin
  Str(AnzahlMessung, s1);
  R.Assign(0, 0, 40, 10);
  DD := New(PDialog, Init(R, mes));
  with DD^ do
  begin
    Options := Options or ofCentered;
    R.Grow(-1, -1);
    Dec(R.B.Y, 3);
    Insert(New(PStaticText, Init(R, #13 + #3 + s + '. von ' + s1
      + ' Messungen' + #13 + #3 + 'Taste <PLOT> druecken' + #13 + #13
      + #3 + 'Abbruch mit <ESC>')));
  end;
  Desktop^.Insert(DD);

  Async_Init;
  if Async_Open(2, 9600, 'N', 8, 1) then
  begin
    FileName := FileName + '.' + ExtPlot;
    Assign(f, FileName);
    ReWrite(f);

    repeat
      received := Async_Buffer_Check(c);
      delay(timeout);
      if keypressed then Taste := readkey;
    until received or (Taste in [chr(27)]);

    if c = chr(17) then
      repeat
        received := Async_Buffer_Check(c);
        delay(timeout);
        if keypressed then Taste := readkey;
      until received or (Taste in [chr(27)]);

    Dispose(DD, Done);
    R.Assign(0, 0, 40, 10);
    DD := New(PDialog, Init(R, 'Messung'));
    with DD^ do
    begin
      Options := Options or ofCentered;
      R.Grow(-1, -1);
```

```
      Dec(R.B.Y, 3);
      Insert(New(PStaticText, Init(R,
        #13 + #3 + 'Daten werden eingelesen ...' + #13 + #13 + #13
        + #3 + 'Abbruch mit <ESC>')));
    end;
    Desktop^.Insert(DD);

    repeat
      Write(f, c);
      received := Async_Buffer_Check(c);
      delay(timeout);
      if keypressed then Taste := readkey;
    until not received or (Taste in [chr(27)]);

    Close(f);
  end;
  Async_Close;
  if Taste = chr(27) then Abbruch := true;
  Dispose(DD, Done);
end; { ReadOut }

{---------------------------------------------------------------------}

{ Dialog zur Messung bei einer Wegdifferenz }
procedure EineMessung;

var
  R                                       : TRect;
  D                                       : PDialog;
  Bruce                                   : PView;
  i, Messung, Command, ad, ad1, am, fnl : word;
  s1                                      : string;
  uebernehmen                             : array [0..2] of boolean;

begin
  repeat
    if Anzahldelta > 0 then ad1 := Anzahldelta - 1
    else
    begin
      ad1 := 0;
      df[0].int := 'i0';
      df[0].obj := 'o0';
      df[0].ref := 'r0';
      df[0].dark := 'd0';
```

```
    AnzahlMessung := 1;
    w[0].delta := 0;
  end;
  ad := Anzahldelta;

  { Parameter der letzten Messung }
  Messdaten.files[0] := df[ad1].int;
  Messdaten.files[1] := df[ad1].obj;
  Messdaten.files[2] := df[ad1].ref;
  Messdaten.files[3] := df[ad1].dark;
  Str(AnzahlMessung : 3, Messdaten.dates[0]);
  Str(w[ad1].delta : 9 : 3, Messdaten.dates[1]);

  { Eingabefenster }
  R.Assign(3, 4, 77, 19);
  D := New(PDialog,Init(R,'Messung'));
  D^.HelpCtx := hcMessung;
  with D^ do
  begin

    { Felder fuer Dateinamen}
    s1 := 'Dateinamen:';
    R.Assign(3, 2, 4 + Length(s1), 3);
    Insert(New(PStaticText, Init(R, s1)));

    R.Assign(23, 4, 33, 5);
    Bruce := New(PInputLine, Init(R, 8));
    Insert(Bruce);
    s1 := 'Interferenzmessung';
    R.Assign(2, 4, 3 + Length(s1), 5);
    Insert(New(PLabel, Init(R, s1, Bruce)));

    R.Assign(23, 6, 33, 7);
    Bruce := New(PInputLine, Init(R, 8));
    Insert(Bruce);
    s1 := 'Objektstrahl';
    R.Assign(2, 6, 3 + Length(s1), 7);
    Insert(New(PLabel, Init(R, s1, Bruce)));

    R.Assign(23, 8, 33, 9);
    Bruce := New(PInputLine, Init(R, 8));
    Insert(Bruce);
    s1 := 'Referenzstrahl';
    R.Assign(2, 8, 3 + Length(s1), 9);
```

```
Insert(New(PLabel, Init(R, s1, Bruce)));

R.Assign(23, 10, 33, 11);
Bruce := New(PInputLine, Init(R, 8));
Insert(Bruce);
s1 := 'Nullintensitaet';
R.Assign(2, 10, 3 + Length(s1), 11);
Insert(New(PLabel, Init(R, s1, Bruce)));

{ Anzahl der Messungen }
R.Assign(61, 4, 71, 5);
Bruce := New(PInputLine, Init(R, 8));
Insert(Bruce);
s1 := 'Anzahl der Messungen';
R.Assign(38, 4, 39 + Length(s1), 5);
Insert(New(PLabel, Init(R, s1, Bruce)));

{ Wegdifferenz delta }
R.Assign(61, 8, 71, 9);
Bruce := New(PInputLine, Init(R, 8));
Insert(Bruce);
s1 := 'Wegdifferenz (cm)';
R.Assign(38, 8, 39 + Length(s1), 9);
Insert(New(PLabel, Init(R, s1, Bruce)));

{ Buttons }
R.Assign(22, 12, 34, 14);
Insert(New(PButton, Init(R, '~M~essung', cmOk, bfDefault)));
R.Assign(40, 12, 52, 14);
Insert(New(PButton, Init(R, '~E~nde', cmCancel, bfNormal)));

{ Messdaten einfuegen }
SetData(Messdaten);

Command := DeskTop^.ExecView(D);
if Command <> cmCancel then
begin
  { Messdaten einlesen }
  GetData(Messdaten);
  Ok := true;
  am := AnzahlMessung;
  df[ad].int := Messdaten.files[0];
  df[ad].obj := Messdaten.files[1];
  df[ad].ref := Messdaten.files[2];
```

```
df[ad].dark := Messdaten.files[3];
ValueI(Messdaten.dates[0], AnzahlMessung);
ValueR(Messdaten.dates[1], w[ad].delta);
Ok := Ok and (AnzahlMessung > 0) and (w[ad].delta >= 0);
if Anzahldelta > 0 then
  if AnzahlMessung <> am then Ok := false;
AnzahlMessung := am;
if Ok then
begin
  if df[ad].int = df[ad].obj then Ok := false;
  if df[ad].int = df[ad].ref then Ok := false;
  if df[ad].int = df[ad].dark then Ok := false;
  if df[ad].obj = df[ad].ref then Ok := false;
  if df[ad].obj = df[ad].dark then Ok := false;
  if df[ad].ref = df[ad].dark then Ok := false;
end;
if Ok then
begin
  case AnzahlMessung of
    1     : fnl := 0;
    2..10 : fnl := 2;
    else fnl := 3;
  end;
  if Length(df[ad].int) > 8 - fnl then Ok := false;
  if Length(df[ad].obj) > 8 - fnl then Ok := false;
  if Length(df[ad].ref) > 8 - fnl then Ok := false;
  if Length(df[ad].dark) > 8 - fnl then Ok := false;
end;
if Ok then
  if Exists(Path + df[ad].int + '.' + ExtPlot) or
    Exists(Path + df[ad].int + '_0' + '.' + ExtPlot)
    then Ok := false;
if Ok then
begin
  for i := 0 to 2 do uebernehmen[i] := false;
  if Exists(Path + df[ad].obj + '.' + ExtPlot) or
    Exists(Path + df[ad].obj + '_0' + '.' + ExtPlot)
    then
    begin
      Ok := false;
      uebernehmen[0] := true;
    end;
  if uebernehmen[0] then if MessageBox(#3 +
    'Objekstrahlmessung existiert' + #13 + #13 + #3 +
```

```
      'Daten uebernehmen ?', nil, mfConfirmation or mfOkCancel)
      = cmOk then Ok := true;
  end;
  if Ok then
  begin
    if Exists(Path + df[ad].ref + '.' + ExtPlot) or
      Exists(Path + df[ad].ref + '_0' + '.' + ExtPlot)
      then
      begin
        Ok := false;
        uebernehmen[1] := true;
      end;
    if uebernehmen[1] then if MessageBox(#3 +
      'Referenzstrahlmessung existiert' + #13 + #13 + #3 +
      'Daten uebernehmen ?', nil, mfConfirmation or mfOkCancel)
      = cmOk then Ok := true;
  end;
  if Ok then
  begin
    if Exists(Path + df[ad].dark + '.' + ExtPlot) or
      Exists(Path + df[ad].dark + '_0' + '.' + ExtPlot)
      then
      begin
        Ok := false;
        uebernehmen[2] := true;
      end;
    if uebernehmen[2] then if MessageBox(#3 +
      'Dunkelmessung existiert' + #13 + #13 + #3 +
      'Daten uebernehmen ?', nil, mfConfirmation or mfOkCancel)
      = cmOk then Ok := true;
  end;
end
else Ok := false;

if (Command = cmOk) and Ok then
begin
  Abbruch := false;
  mes := 'Interferenzmessung';
  for Messung := 0 to AnzahlMessung - 1 do
  begin
    FileName := Path + df[ad].int;
    if AnzahlMessung > 1 then
    begin
      Str(Messung, s);
```

```
      FileName := FileName + '_' + s;
      Str(Messung + 1, s);
    end
    else s := '1';
    ReadOut;
    if Abbruch then Messung := AnzahlMessung - 1;
  end;
  mes := 'Objektstrahl';
  if not uebernehmen[0] and not Abbruch then
  begin
    for Messung := 0 to AnzahlMessung - 1 do
    begin
      FileName := Path + df[ad].obj;
      if AnzahlMessung > 1 then
      begin
        Str(Messung, s);
        FileName := FileName + '_' + s;
        Str(Messung + 1, s);
      end
      else s := '1';
      ReadOut;
      if Abbruch then Messung := AnzahlMessung - 1;
     end;
  end;
  mes := 'Referenzstrahl';
  if not uebernehmen[1] and not Abbruch then
  begin
    for Messung := 0 to AnzahlMessung - 1 do
    begin
      FileName := Path + df[ad].ref;
      if AnzahlMessung > 1 then
      begin
        Str(Messung, s);
        FileName := FileName + '_' + s;
        Str(Messung + 1, s);
      end
      else s := '1';
      ReadOut;
      if Abbruch then Messung := AnzahlMessung - 1;
    end;
  end;
  mes := 'Nullintensitaet';
  if not uebernehmen[2] and not Abbruch then
  begin
```

```
          for Messung := 0 to AnzahlMessung - 1 do
          begin
            FileName := Path + df[ad].dark;
            if AnzahlMessung > 1 then
            begin
              Str(Messung, s);
              FileName := FileName + '_' + s;
              Str(Messung + 1, s);
            end
            else s := '1';
            ReadOut;
            if Abbruch then Messung := AnzahlMessung - 1;
          end;
        end;
        if not Abbruch then inc(Anzahldelta);
      end;
      if (Command = cmOk) and not Ok then
        MessageBox(#3 + 'Eingabefehler:' + #13 + #3
          + 'Dateiname existiert, doppelt,' + #13 + #3
          + 'zu lang ? Werte ungueltig ?' + #13 + #3
          + 'Anzahl der Messungen veraendert ?',
          nil, mfError or mfOkButton);
    end;
    Dispose(D, Done);
  until (Command = cmCancel);
end; { EineMessung }

{--------------------------------------------------------------------}

{ Dialog zur Auswahl der Protokolldatei }
procedure Messdialog;

var
  R       : TRect;
  D       : PFileDialog;
  Command : word;
  fnsave  : string;

begin
  Extens := '*.' + ExtDaten;
  Ok := false;
  repeat
    D := New(PFileDialog, Init(Extens, 'Protokolldatei',
      '~N~ame (*.DAT)', fdOkButton, fdReplaceButton));
```

```
    Command := Desktop^.ExecView(D);
    if Command <> cmCancel then
    begin
      D^.GetFileName(FileName);
      if Exists(FileName) then
      begin
        if MessageBox(#3 + 'Datei existiert !' + #13 + #3
          + 'Messungen zufuegen ?', nil, mfInformation or mfOkCancel)
          = cmOk then
        begin
          LoadDatFile;
          Ok := true;
        end;
      end
      else
      begin
        Anzahldelta := 0;
        Ok := true
      end;
    end
    else Ok:= true;
    Dispose(D, Done);
  until Ok = true;
  if Command <> cmCancel then
  begin
    fnsave := FileName;
    Path := copy(FileName, 0, Length(FileName) - 4);
    repeat
      Path := copy(Path, 0, Length(Path) - 1);
      s := copy(Path, Length(Path), Length(Path));
    until s = '\';
    repeat
      EineMessung;
    until MessageBox(#13 + #3 + 'weitere Messung ?', nil,
      mfConfirmation or mfOkCancel) = cmCancel;
    FileName := fnsave;
    if Anzahldelta > 0 then SaveDatFile;
  end;
end; { Messdialog }

{-------------------------------------------------------------------}

end. { MessU }
```

```
{--------------------------------------------------------------------}
```

D.4 Unit ASync4U1

```
{--------------------------------------------------------------------}
{                          ASYNC4U.PAS                               }
{                                                                    }
{  This is a faithful translation of the famous ASYNC.INC by         }
{  Michael Quinlan into a Turbo 4.0 unit.  No extra frills, no       }
{  modification of types, nothing fancy.  But with this code you     }
{  should be able to delete your $I ASYNC.INC directive, add a USES }
{  aASYNC4U statement, nd recompile your existing program.  If you  }
{  want to add support for more ports, other computers, or change    }
{  to use the new data types, all good ideas, go right ahead. With  }
{  this you don't have to.                                           }
{                                                                    }
{                                 Scott Gurvey, November 29 1987   }
{--------------------------------------------------------------------}
{                                                                    }
{                           ASYNC.INC                                }
{                                                                    }
{  Async Communication Routines                                      }
{  by Michael Quinlan                                                }
{  with a bug fixed by Scott Herr                                    }
{  made PCjr-compatible by W. M. Miller                              }
{  Highly dependant on the IBM PC and PC DOS 2.0                     }
{                                                                    }
{  based on the DUMBTERM program by CJ Dunford in the January 1984  }
{  issue of PC Tech Journal.                                         }
{                                                                    }
{  Entry points:                                                     }
{                                                                    }
{    Async_Init                                                      }
{      Performs initialization.                                      }
{                                                                    }
{    Async_Open(Port, Baud : Integer;                                }
{               Parity : Char;                                       }
{               WordSize, StpBits : Integer) : Boolean               }
{      Sets up interrupt vector, initialies the COM port for         }
{      processing, sets pointers to the buffer.  Returns FALSE if    }
{      COM port not installed.                                       }
{                                                                    }
{    Async_Buffer_Check(var C : Char) : Boolean                      }
```

```
{      If a character is available, returns TRUE and moves the      }
{        character from the buffer to the parameter                 }
{      Otherwise, returns FALSE                                     }
{                                                                   }
{    Async_Send(C : Char)                                           }
{      Transmits the character.                                     }
{                                                                   }
{    Async_Send_String(S : LStr)                                    }
{      Calls Async_Send to send each character of S.                }
{                                                                   }
{    Async_Close                                                    }
{      Turn off the COM port interrupts.                            }
{      **MUST** BE CALLED BEFORE EXITING YOUR PROGRAM; otherwise    }
{      you will see some really strange errors and have to re-boot. }
{                                                                   }
{-------------------------------------------------------------------}

{$B-} { Short circuit boolean ON }
{$I-} { I/O hecking OFF }
{$R-} { Range checking OFF }
{$S-} { Stack checking OFF }
{$V-} { Var-str checking OFF}

unit ASync4U1;

interface

uses Dos;

{ global declarations }

type
  LStr = String[255];  { generic string type for parameters }

const
  Async_Buffer_Max = 4095;

var
  Async_OriginalVector : pointer;
  Async_Buffer         : Array[0..Async_Buffer_Max] of char;

  Async_Open_Flag : Boolean;    { true if Open but no Close }
  Async_Port      : Integer;    { current Open port number (1 or 2) }
  Async_Base      : Integer;    { base for current open port }
```

```
  Async_Irq       : Integer;    { irq for current open port }

  Async_Buffer_Overflow : Boolean;
                            { True if buffer overflow has happened }
  Async_Buffer_Used     : Integer;
  Async_MaxBufferUsed   : Integer;

    { Async_Buffer is empty if Head = Tail }
  Async_Buffer_Head    : Integer;
                           { Locn in Async_Buffer to put next char }
  Async_Buffer_Tail    : Integer;
                           { Locn in Async_Buffer to get next char }
  Async_Buffer_NewTail : Integer;

{--------------------------------------------------------------------}
{                      USER CALLABLE ROUTINES                        }
{--------------------------------------------------------------------}

procedure Async_Init;
{ initialize variables }

procedure Async_Close;
{ reset the interrupt system when UART interrupts no longer needed }

function Async_Open(ComPort  : Integer;
                    BaudRate : Integer;
                    Parity   : Char;
                    WordSize : Integer;
                    StopBits : Integer) : Boolean;
{ open a communications port }

function Async_Buffer_Check(var C : Char) : Boolean;
{ see if a character has been received; return it if yes }

procedure Async_Send(C : Char);
{ transmit a character }

procedure Async_Send_it(C : integer);
{ trancmit a integer }

procedure Async_Send_String(S : LStr);
{ transmit a string }

{--------------------------------------------------------------------}
```

```
implementation

const
  UART_THR = $00;   { offset from base of UART Registers for IBM PC }
  UART_RBR = $00;
  UART_IER = $01;
  UART_IIR = $02;
  UART_LCR = $03;
  UART_MCR = $04;
  UART_LSR = $05;
  UART_MSR = $06;

  I8088_IMR = $21;  { port address of the Interrupt Mask Register }

var
  Async_BIOS_Port_Table : Array[1..2] of Integer absolute $40:0;
          { This table is initialized by BIOS equipment determination
            code at boot time to contain the base addresses for the
            installed async adapters.  A value of 0 means "not in-
            stalled." }

const
  Async_Num_Bauds = 8;
  Async_Baud_Table : array [1..Async_Num_Bauds] of record
                       Baud, Bits : integer
                     end
                   = ((Baud:110;  Bits:$00),
                      (Baud:150;  Bits:$20),
                      (Baud:300;  Bits:$40),
                      (Baud:600;  Bits:$60),
                      (Baud:1200; Bits:$80),
                      (Baud:2400; Bits:$A0),
                      (Baud:4800; Bits:$C0),
                      (Baud:9600; Bits:$E0));

procedure DisableInterrupts; inline($FA {cli} );     {MACROS}
procedure EnableInterrupts;  inline($FB {sti} );

{-------------------------------------------------------------------}

procedure BIOS_RS232_Init(ComPort, ComParm : Integer);
{ Issue Interrupt $14 to initialize the UART }
{ See the IBM PC Technical Reference Manual for the format of
```

```
  ComParm }

var
  Regs : registers;

begin
  with Regs do
    begin
      ax := ComParm and $00FF;  { AH=0; AL=ComParm }
      dx := ComPort;
      Intr($14, Regs)
    end
end; { BIOS_RS232_Init }

{---------------------------------------------------------------------}
{                                                                     }
{  ISR - Interrupt Service Routine                                    }
{                                                                     }
{---------------------------------------------------------------------}

procedure Async_Isr;  interrupt;
{ Interrupt Service Routine }
{ Invoked when the UART has received a byte of data from the
  communication line }

{ re-written 9/10/84 to be entirely in machine language; original
  source left as comments }

begin
  Inline(
    $FB/                           { STI }
      { get the incomming character }
      { Async_Buffer[Async_Buffer_Head] :=
          Chr(Port[UART_RBR + Async_Base]); }
    $8B/$16/Async_Base/            { MOV DX,Async_Base }
    $EC/                           { IN AL,DX }
    $8B/$1E/Async_Buffer_Head/     { MOV BX,Async_Buffer_Head }
    $88/$87/Async_Buffer/          { MOV Async_Buffer[BX],AL }
      { Async_Buffer_NewHead := Async_Buffer_Head + 1; }
    $43/                           { INC BX }
      { if Async_Buffer_NewHead > Async_Buffer_Max then
          Async_Buffer_NewHead := 0; }
    $81/$FB/Async_Buffer_Max/      { CMP BX,Async_Buffer_Max }
    $7E/$02/                       { JLE L001 }
```

```
    $33/$DB/                      { XOR BX,BX }
      { if Async_Buffer_NewHead = Async_Buffer_Tail then
          Async_Buffer_Overflow := TRUE
        else }
{L001:}
    $3B/$1E/Async_Buffer_Tail/    { CMP BX,Async_Buffer_Tail }
    $75/$08/                      { JNE L002 }
    $C6/$06/Async_Buffer_Overflow/$01/
                                  { MOV Async_Buffer_Overflow,1 }
    $90/                          { NOP generated by assembler for
                                    some reason }
    $EB/$16/                      { JMP SHORT L003 }
      { begin
          Async_Buffer_Head := Async_Buffer_NewHead;
          Async_Buffer_Used := Async_Buffer_Used + 1;
          if Async_Buffer_Used > Async_MaxBufferUsed then
            Async_MaxBufferUsed := Async_Buffer_Used
        end; }
{L002:}
    $89/$1E/Async_Buffer_Head/    { MOV Async_Buffer_Head,BX }
    $FF/$06/Async_Buffer_Used/    { INC Async_Buffer_Used }
    $8B/$1E/Async_Buffer_Used/    { MOV BX,Async_Buffer_Used }
    $3B/$1E/Async_MaxBufferUsed/  { CMP BX,Async_MaxBufferUsed }
    $7E/$04/                      { JLE L003 }
    $89/$1E/Async_MaxBufferUsed/  { MOV Async_MaxBufferUsed,BX }
{L003:}
      { disable interrupts }
    $FA/                          { CLI }
      { Port[$20] := $20; }  { use non-specific EOI }
    $B0/$20/                      { MOV AL,20h }
    $E6/$20                       { OUT 20h,AL }
       )
end; { Async_Isr }

{--------------------------------------------------------------------}

procedure Async_Init;
{ initialize variables }

begin
  Async_Open_Flag := FALSE;
  Async_Buffer_Overflow := FALSE;
  Async_Buffer_Used := 0;
  Async_MaxBufferUsed := 0;
```

```
end; { Async_Init }

{-----------------------------------------------------------------------}

procedure Async_Close;
{ reset the interrupt system when UART interrupts no longer needed }

var
  i, m : Integer;

begin
  if Async_Open_Flag then
    begin
      { disable the IRQ on the 8259 }
      DisableInterrupts;
      i := Port[I8088_IMR];       { get the interrupt mask register }
      m := 1 shl Async_Irq;       { set mask to turn off interrupt }
      Port[I8088_IMR] := i or m;

      { disable the 8250 data ready interrupt }
      Port[UART_IER + Async_Base] := 0;

      { disable OUT2 on the 8250 }
      Port[UART_MCR + Async_Base] := 0;
      EnableInterrupts;

      SetIntVec(Async_Irq + 8,Async_OriginalVector);

      { re-initialize our data areas so we know the port is closed }
      Async_Open_Flag := FALSE
    end
end; { Async_Close }

{-----------------------------------------------------------------------}

function Async_Open(ComPort  : Integer;
                    BaudRate : Integer;
                    Parity   : Char;
                    WordSize : Integer;
                    StopBits : Integer) : Boolean;
{ open a communications port }

var
  ComParm : Integer;
```

```
  i, m : Integer;

begin
  if Async_Open_Flag then Async_Close;

  if (ComPort = 2) and (Async_BIOS_Port_Table[2] <> 0) then
    Async_Port := 2
  else
    Async_Port := 1;  { default to COM1 }
  Async_Base := Async_BIOS_Port_Table[Async_Port];
  Async_Irq := Hi(Async_Base) + 1;

  if (Port[UART_IIR + Async_Base] and $00F8) <> 0 then
    Async_Open := FALSE
  else
    begin
      Async_Buffer_Head := 0;
      Async_Buffer_Tail := 0;
      Async_Buffer_Overflow := FALSE;

  { Build the ComParm for RS232_Init }
  { See Technical Reference Manual for description }

      ComParm := $0000;

  { Set up the bits for the baud rate }
      i := 0;
      repeat
        i := i + 1
      until (Async_Baud_Table[i].Baud = BaudRate) or
        (i = Async_Num_Bauds);
      ComParm := ComParm or Async_Baud_Table[i].Bits;

      if Parity in ['E', 'e'] then ComParm := ComParm or $0018
      else if Parity in ['O', 'o'] then ComParm := ComParm or $0008
      else ComParm := ComParm or $0000;  { default to No parity }

      if WordSize = 7 then ComParm := ComParm or $0002
      else ComParm := ComParm or $0003;  { default to 8 data bits }

      if StopBits = 2 then ComParm := ComParm or $0004
      else ComParm := ComParm or $0000;  { default to 1 stop bit }

  { use the BIOS COM port initialization routine to save typing the
```

```
    code }
      BIOS_RS232_Init(Async_Port - 1, ComParm);

      GetIntVec(Async_Irq + 8, Async_OriginalVector);
      SetIntVec(Async_Irq + 8, @Async_Isr);

  { read the RBR and reset any possible pending error conditions }
  { first turn off the Divisor Access Latch Bit to allow access to
    RBR, etc. }

      DisableInterrupts;

      Port[UART_LCR + Async_Base] :=
        Port[UART_LCR + Async_Base] and $7F;
  { read the Line Status Register to reset any errors it indicates }
      i := Port[UART_LSR + Async_Base];
  { read the Receiver Buffer Register in case it contains a
    character }
      i := Port[UART_RBR + Async_Base];

  { enable the irq on the 8259 controller }
      i := Port[I8088_IMR];  { get the interrupt mask register }
      m := (1 shl Async_Irq) xor $00FF;
      Port[I8088_IMR] := i and m;

  { enable the data ready interrupt on the 8250 }
      Port[UART_IER + Async_Base] := $01;
                             { enable data ready interrupt }

  { enable OUT2 ****und DTR**** on 8250 }
      i := Port[UART_MCR + Async_Base];
      Port[UART_MCR + Async_Base] := i or $09; { ****$08**** }

      EnableInterrupts;
      Async_Open_Flag := TRUE;  { bug fix by Scott Herr }
      Async_Open := TRUE
    end
end; { Async_Open }

{---------------------------------------------------------------------}

function Async_Buffer_Check(var C : Char) : Boolean;
{ see if a character has been received; return it if yes }
```

```
var
  i : word;

begin
  { **** }
  { enable OUT2 ****und DTR**** on 8250 }
      i := Port[UART_MCR + Async_Base];
      Port[UART_MCR + Async_Base] := i or $09;
  { **** }

  if Async_Buffer_Head = Async_Buffer_Tail then
    Async_Buffer_Check := FALSE
  else
    begin
      C := Async_Buffer[Async_Buffer_Tail];
      Async_Buffer_Tail := Async_Buffer_Tail + 1;
      if Async_Buffer_Tail > Async_Buffer_Max then
        Async_Buffer_Tail := 0;
      Async_Buffer_Used := Async_Buffer_Used - 1;
      Async_Buffer_Check := TRUE
    end
end; { Async_Buffer_Check }

{--------------------------------------------------------------------}

procedure Async_Send(C : Char);
{ transmit a character }

var
  i, m, counter : Integer;

begin
  Port[UART_MCR + Async_Base] := $0B; { turn on OUT2, DTR, and RTS }

  { wait for CTS }
  counter := MaxInt;
  while (counter <> 0) and
    ((Port[UART_MSR + Async_Base] and $10) = 0) do
    counter := counter - 1;

  { wait for Transmit Hold Register Empty (THRE) }
  if counter <> 0 then counter := MaxInt;
  while (counter <> 0) and
    ((Port[UART_LSR + Async_Base] and $20) = 0) do
```

```
    counter := counter - 1;

  if counter <> 0 then
    begin
      { send the character }
      DisableInterrupts;
      Port[UART_THR + Async_Base] := Ord(C);
      EnableInterrupts
    end
  else
    writeln('<<<TIMEOUT>>>');
end; { Async_Send }

{-------------------------------------------------------------------}

procedure Async_Send_it(C : integer);
{ transmit a integer }

var
  i, m, counter : Integer;

begin
  Port[UART_MCR + Async_Base] := $0B; { turn on OUT2, DTR, and RTS }

  { wait for CTS }
  counter := MaxInt;
  while (counter <> 0) and
    ((Port[UART_MSR + Async_Base] and $10) = 0) do
    counter := counter - 1;

  { wait for Transmit Hold Register Empty (THRE) }
  if counter <> 0 then counter := MaxInt;
  while (counter <> 0) and
    ((Port[UART_LSR + Async_Base] and $20) = 0) do
    counter := counter - 1;

  if counter <> 0 then
    begin
      { send the character }
      DisableInterrupts;
      Port[UART_THR + Async_Base] := Ord(C);
      EnableInterrupts
    end
  else
```

```
    writeln('<<<TIMEOUT>>>');
end; { Async_Send_it }

{------------------------------------------------------------------}

procedure Async_Send_String(S : LStr);
{ transmit a string }

var
  i : Integer;

begin
  for i := 1 to length(S) do
    Async_Send(S[i])
end; { Async_Send_String }

{------------------------------------------------------------------}

end. { ASYNC4U1 }

{------------------------------------------------------------------}
```

D.5 Unit ConvertU

```
{------------------------------------------------------------------}

{ Konvertierung einer Plot-Datei in Integer-Werte }
unit ConvertU;

{------------------------------------------------------------------}

interface

procedure Convert;
procedure ConvertFile;

{------------------------------------------------------------------}

implementation

uses
  Glob,         { Globale Deklarationen }
  LoadSave,     { Laden und Speichern von Dateien }
```

```
  { Turbo Pascal Standard Units }
  App, Crt, Dialogs, MsgBox, Objects, StdDlg, Views;

{--------------------------------------------------------------------}

{ Ermitteln der Integer-Werte }
procedure Convert;

type
  Wert = record
    x, y : word;
  end;

var
  w               : array [0..nmax - 1] of Wert;
  i, j, ErrorCode : word;
  s               : string [5];
  c               : char;
  f               : Text;

begin
  Assign(f, FileName);
  Reset(f);
  repeat
    Read(f, c);
  until c = 'D';
  repeat
    Read(f, c);
  until c = 'D';
  Read(f, c); { ';' }
  Read(f, c); { 'P' }
  Read(f, c); { 'A' }
  i := 0;
  repeat
    Read(f, c); { ' ' }
    s := '';
    c := ' ';
    repeat
      s := s + c;
      Read(f, c);
    until c = ',';
    Val(s, w[i].x, ErrorCode);
    w[i].x := w[i].x div 4;
```

```
    s := '';
    c := ' ';
    repeat
      s := s + c;
      Read(f, c);
    until c = ';';
    Val(s, w[i].y, ErrorCode);
    w[i].y := w[i].y div 16 - 22;
    Read(f, c); { 'P' }
    Read(f, c); { 'A' bzw. 'U' }
    inc(i);
  until c = 'U';
  Close(f);

  i := 0;
  Imes^[0] := w[0].y;
  for j := 1 to nmax - 1 do
  begin
    if w[i].x > j then Imes^[j] := Imes^[j - 1]
    else
    begin
      Imes^[j] := w[i].y;
      inc(i);
    end;
  end;
  FileName := copy(FileName, 0, length(FileName) - 3) + ExtMess;
  SaveFileI(Imes^, FileName);
end; { Convert }

{--------------------------------------------------------------------}

{ Konvertierungs-Dialog }
procedure ConvertFile;

var
  D       : PFileDialog;
  Command : word;

begin
  Extens := '*' + ExtPlot;
  repeat
    D := New(PFileDialog, Init(Extens, 'Datei oeffnen',
      '~N~ame (*.PLT)', fdOpenButton, fdReplaceButton));
    Command := Desktop^.ExecView(D);
```

```
    if (Command <> cmCancel) then
    begin
      D^.GetFileName(FileName);
      if Exists(FileName) then Convert
      else MessageBox(#3 + 'Datei ' + FileName + ' nicht gefunden'
        + #13 + #13, nil, mfError or mfOKButton);
    end;
    Dispose(D, Done);
  until (Command = cmCancel);
end; { ConvertFile }

{---------------------------------------------------------------------}

end. { ConvertU }

{---------------------------------------------------------------------}
```

D.6 Unit TransU

```
{---------------------------------------------------------------------}

{ Vorbereitungen zur Fast Fourier Transformation und zur inversen
  Fast Fourier Transformation }
unit TransU;

{---------------------------------------------------------------------}

interface

procedure FFTBerechnung;
procedure Synthese;

{---------------------------------------------------------------------}

implementation

uses
  ConvertU,    { Konvertierung von Plot-Daten }
  FFTU,        { (inverse) FFT ohne Coprozessor }
  Glob,        { Globale Deklarationen }
  LoadSave,    { Laden und Speichern von Dateien }
  XFFT;        { (inverse) FFT mit Coprozessor }
```

```
var
  FFTEnde : word;

{------------------------------------------------------------------}

{ Ermitteln der Signalfrequenz und Abschneiden der FFT }
procedure FFTCut;

var
  i, FFTmin : word;
  tmp       : real;

begin
  i := 1;
  while Itrans^[i] < Itrans^[i - 1] do inc(i);
  FFTmin := i;

  tmp := Itrans^[FFTmin];
  FFTmax := FFTmin;
  for i := FFTmin + 1 to nmax div 2 do
    if Itrans^[i] > tmp then
    begin
      tmp := Itrans^[i];
      FFTmax := i;
    end;

  if FFTmax < nmax div 4 then FFTEnde := round(2 * FFTmax)
  else FFTEnde := 10;
  Tau := nmax / FFTmax;
end; { FFTCut }

{------------------------------------------------------------------}

{ Vor- und Nachbereitungen der (inversen) FFT }
procedure Transformation(var h : boolean);

var
  i : word;

begin
  { Vorbereitung der Werte fuer die (inverse) FFT }
  if h then
    for i := 0 to nmax - 1 do
    begin
```

```
      IRe^[i] := Imes^[i];
      IIm^[i] := 0;
    end
  else
  begin
    FFTCut;
    for i := FFTEnde + 1 to nmax - FFTEnde - 1 do
    begin
      IRe^[i] := 0;
      IIm^[i] := 0;
    end;
  end;

{$IFDEF CPU87}

  { Berechnung der (inversen) FFT ohne Coprozessor ueber FFTU }
  CoSiTab(nmax);
  FFTAsm(IRe^, IIm^, nmax, round(ln(nmax) / ln(2)), h);

{$ELSE}

  { Berechnung der (inversen) FFT mit Coprozessor ueber XFFT }
  Tabelle(nmax);
  FFT(IRe^, IIm^, nmax, round(ln(nmax) / ln(2)), h);

{$ENDIF}

  { Betragsbildung }
  for i := 0 to nmax - 1 do
    Itrans^[i] := sqrt(sqr(IRe^[i]) + sqr(IIm^[i]));
  if h then Itrans^[0] := Itrans^[0] / 2;
end; { Transformation }

{--------------------------------------------------------------------}

{ Aufrufen der FFT }
procedure FFTBerechnung;

var
  FFThin : boolean;

begin
  if not Exists(copy(FileName, 0, Length(FileName) - 3) + ExtMess)
    then Convert
```

```
  else FileName := copy(FileName, 0, Length(FileName) - 3) + ExtMess;
  LoadFileI(Imes^, FileName);
  FFThin := true;
  Transformation(FFThin);
  FileName := copy(FileName, 0, Length(FileName) - 3) + ExtFFT;
end; { FFTBerechnung }

{---------------------------------------------------------------------}

{ Aufrufen der inversen FFT }
procedure Synthese;

var
  FFThin : boolean;

begin
  FFTBerechnung;
  FFThin := false;
  Transformation(FFThin);
  FileName := copy(FileName, 0, Length(FileName) - 3) + ExtKor;
end; { Synthese }

{---------------------------------------------------------------------}

end. { TransU }

{---------------------------------------------------------------------}
```

D.7 Unit FFTU

```
{---------------------------------------------------------------------}

{ Eindimensionale Fast Fourier Transformation ohne Coprozessor }
unit FFTU;

{$N+,E+}

{ Achtung : die Unit FFTU benoetigt waehrend der Kompilation die
            Datei BITREV.OBJ ! }

{---------------------------------------------------------------------}

interface
```

```
uses
  Glob;     { Globale Deklarationen }

procedure Tabelle(n : integer);  { Muss vor dem ersten Aufruf von
                                   FFT aufgerufen werden ! }
procedure FFT(var XReal, XImag : Realarray; n, nu : integer;
              hin : boolean);

{-------------------------------------------------------------------}

implementation

type
  RealTypearray = array [0..nmax - 1] of single;

var
  i, q1, q2, q3, q4 : integer;
  Tabarray          : ^RealTypearray;

{-------------------------------------------------------------------}

{$L BITREV}
{$F-}

function BitRev(m, nu : integer): integer; external;

{$F+}

{-------------------------------------------------------------------}

procedure Tabelle(n : integer);

var
  i, nmax : integer;
  s       : single;

begin
  nmax := n div 4;
  s := 2.0 * Pi / n;
  for i := 0 to nmax do Tabarray^[i] := sin(s * i);
  q1 := n div 4;
  q2:= n div 2;
  q3 := q1 + q2;
```

```
  q4 := n;
end; { Tabelle }

{-------------------------------------------------------------------}

function Sinus(x : integer): single;

begin
  if x <= q1 then Sinus :=  Tabarray^[x]
  else if x <= q2 then Sinus :=  Tabarray^[q2 - x]
       else if x <= q3 then Sinus :=  - Tabarray^[x - q2]
            else Sinus :=  - Tabarray^[q4 - x]
end; { Sinus }

{-------------------------------------------------------------------}

function Cosinus(x : integer): single;

begin
  if x <= q1 then Cosinus :=  Tabarray^[q1 - x]
  else if x <= q2 then Cosinus := - Tabarray^[x - q1]
       else if x <= q3 then Cosinus := - Tabarray^[q3 - x]
            else Cosinus := Tabarray^[x - q3]
end; { Cosinus }

{-------------------------------------------------------------------}

procedure FFT(var XReal, XImag : Realarray; n, nu : integer;
              hin : boolean);

type
  PointRec = record
    Low, High : word;
  end;

var
  n2, kn2, nu1, i, l, k, k0, k0step, p, pstep : integer;
  FFTfilename                                 : string [25];
  f                                           : Text;
  TReal, TImag, c, s, rn2, FFTNorm            : single;
  RealPtr1, ImagPtr1, RealPtr2, ImagPtr2      : ^single;

  RRealPtr1 : PointRec absolute RealPtr1;
  RRealPtr2 : PointRec absolute RealPtr2;
```

```
  RImagPtr1 : PointRec absolute ImagPtr1;
  RImagPtr2 : PointRec absolute ImagPtr2;

begin
  k := 0;

  while (k < n) do
  begin
    i := BitRev(k, nu);
    if (i > k) then
    begin
      TReal := XReal[k];
      TImag := XImag[k];
      XReal[k] := XReal[i];
      XImag[k] := XImag[i];
      XReal[i] := TReal;
      XImag[i] := TImag
    end;
    inc(k)
  end;

  n2 := 1;
  nu1 := 1;
  k0step := 2;
  pstep := n shr 1;

  for l := 1 to nu do
  begin
    k0 := 0;

    repeat
      k := k0;
      kn2 := k0 + n2;
      p := 0;
      RealPtr1 := @XReal[kn2];
      ImagPtr1 := @XImag[kn2];
      RealPtr2 := @XReal[k];
      ImagPtr2 := @XImag[k];
      for i := 1 to n2 do
      begin
        c := Cosinus(p);
        if hin then s := - Sinus(p) else s := Sinus(p);

        TReal := RealPtr1^ * c + ImagPtr1^ * s;
```

```
        TImag := ImagPtr1^ * c - RealPtr1^ * s;

        RealPtr1^ := RealPtr2^ - TReal;
        ImagPtr1^ := ImagPtr2^ - TImag;
        RealPtr2^ := RealPtr2^ + TReal;
        ImagPtr2^ := ImagPtr2^ + TImag;

        inc(RRealPtr1.Low, sizeof(Single));
        inc(RRealPtr2.Low, sizeof(Single));
        inc(RImagPtr1.Low, sizeof(Single));
        inc(RImagPtr2.Low, sizeof(Single));

        inc(p, pstep)
      end;
      inc(k0, k0step)
    until (k0 = n);

    k0step := k0step + k0step;
    inc(nu1);
    n2 := n2 + n2;
    pstep := pstep shr 1
  end;

  if hin then
  begin
    rn2 := 2 / n;
    for i := 0 to n - 1 do
    begin
      XReal[i] := rn2 * XReal[i];
      XImag[i] := rn2 * XImag[i];
    end;
  end
  else
  begin
    for i := 0 to n - 1 do
    begin
      XReal[i] := 0.5 * XReal[i];
      XImag[i] := 0.5 * XImag[i];
    end;
  end;
end; { FFT }

{------------------------------------------------------------------}
```

```
end. { FFTU }

{-----------------------------------------------------------------------}
```

D.8 Unit XFFT

```
{-----------------------------------------------------------------------}

{ Eindimensionale Fast Fourier Transformation mit Coprozessor }
unit XFFT;

{$N+,E+}

{ Achtung : die Unit XFFT benoetigt waehrend der Kompilation die
            Datei FASM.OBJ ! }

{-----------------------------------------------------------------------}

interface

uses
  Glob;      { Globale Deklarationen }

{-----------------------------------------------------------------------}

procedure CoSiTab(n : integer); { Muss vor dem ersten Aufruf von
                                  FFTAsm aufgerufen werden ! }
procedure FFTAsm(var xr, xi : Realarray; n, nu : integer;
          hin : boolean);

{-----------------------------------------------------------------------}

implementation

var
  CosTab : array [0..nmax - 1] of single;
  SinTab : array [0..nmax - 1] of single;

{-----------------------------------------------------------------------}

procedure CoSiTab(n : integer);

var
```

```
  i : integer;
  s : single;

begin
  s := 2 * Pi / n;
  for i := 0 to n - 1 do
  begin
    SinTab[i] := sin(s * i);
    CosTab[i] := cos(s * i);
  end;
end; { CoSiTab }

{-------------------------------------------------------------------}

procedure FFTAsm(var xr, xi : Realarray; n, nu : integer;
          hin : boolean); external;

{$L FASM}

{-------------------------------------------------------------------}

end. { XFFT }

{-------------------------------------------------------------------}
```

D.9 Unit KoherU

```
{-------------------------------------------------------------------}

{ Berechnung des Kohaerenzgrades und der Kohaerenzlaenge }
unit KoherU;

{-------------------------------------------------------------------}

interface

procedure gammaBerechnung;
procedure LCBerechnung;

{-------------------------------------------------------------------}

implementation
```

```
uses
  Glob,         { Globale Deklarationen }
  LoadSave,     { Laden und Speichern von Dateien }
  TransU,       { Vorbereitungen zur (inversen) FFT }

  { Turbo Pascal Standard Units }
  App, Crt, Dialogs, Objects, Views;

{--------------------------------------------------------------------}

{ Berechnung des Kohaerenzgrades }
procedure gammaBerechnung;

var
  R                                                    : TRect;
  DD                                                   : PDialog;
  i, j, Messung, max                                   : word;
  I0, I0n, Imax, Imin, Imin1, M, gam, Summe, QSumme : real;
  Iref, Iobj                                           : ^Realarray;
  fnsave, s, s1                                        : string;
  Taste                                                : char;

begin
  New(Iref);
  New(Iobj);
  Taste := ' ';
  fnsave := FileName;
  Path := copy(FileName, 0, Length(FileName) - 4);
  repeat
    Path := copy(Path, 0, Length(Path) - 1);
    s := copy(Path, Length(Path), Length(Path));
  until s = '\';

  for j := 0 to Anzahldelta - 1 do
  begin
    Str(j + 1, s);
    Str(Anzahldelta, s1);
    R.Assign(0, 0, 40, 10);
    DD := New(PDialog, Init(R, 'Berechnung'));
    with DD^ do
    begin
      Options := Options or ofCentered;
      R.Grow(-1, -1);
      Dec(R.B.Y, 3);
```

```
    Insert(New(PStaticText, Init(R, #13 + #3 + s + '. von ' + s1
      + ' Abstaenden' + #13 + #13 + #13 + #3
      + 'Abbruch mit <ESC>')));
  end;
  Desktop^.Insert(DD);
  I0 := 0;
  for i := 0 to nmax - 1 do
  begin
    Iref^[i] := 0;
    Iobj^[i] := 0;
  end;

  { Nullintensitaet }
  for Messung := 0 to AnzahlMessung - 1 do
  begin
    FileName := Path + df[j].dark;
    if AnzahlMessung > 1 then
    begin
      Str(Messung, s);
      FileName := FileName + '_' + s;
    end;
    FileName := FileName + '.' + ExtMess;
    LoadFileI(Imes^, FileName);
    I0n := 0;
    for i := 0 to nmax - 1 do I0n := I0n + Imes^[i];
    I0n := I0n / nmax;
    I0 := (I0n + Messung * I0) / (Messung + 1);
  end;

  { Referenzstrahl }
  for Messung := 0 to AnzahlMessung - 1 do
  begin
    FileName := Path + df[j].ref;
    if AnzahlMessung > 1 then
    begin
      FileName := FileName + '_' + s;
      Str(Messung, s);
    end;
    FileName := FileName + '.' + ExtPlot;
    Synthese;
    for i := 0 to nmax - 1 do
    begin
      Itrans^[i] := Itrans^[i] - I0;
      Iref^[i] :=
```

```
      Iref^[i] + (Itrans^[i] - Iref^[i]) / (Messung + 1);
  end;
end;

{ Objektstrahl }
for Messung := 0 to AnzahlMessung - 1 do
begin
  FileName := Path + df[j].obj;
  if AnzahlMessung > 1 then
  begin
    FileName := FileName + '_' + s;
    Str(Messung, s);
  end;
  FileName := FileName + '.' + ExtPlot;
  Synthese;
  for i := 0 to nmax - 1 do
  begin
    Itrans^[i] := Itrans^[i] - I0;
    Iobj^[i] :=
      Iobj^[i] + (Itrans^[i] - Iobj^[i]) / (Messung + 1);
  end;
end;

{ Interferenzmuster }
w[j].gamma := 0;
Summe := 0;
QSumme := 0;
for Messung := 0 to AnzahlMessung - 1 do
begin
  FileName := Path + df[j].int;
  if AnzahlMessung > 1 then
  begin
    FileName := FileName + '_' + s;
    Str(Messung, s);
  end;
  FileName := FileName + '.' + ExtPlot;
  Synthese;

  { Maximalintensitaet }
  Imax := 0;
  for i := 0 to nmax - 1 do
    if Itrans^[i] > Imax then
    begin
      Imax := Itrans^[i];
```

```
    max := i;
  end;

{ Minimalintensitaet }
Imin := Imax;
if (max > 0.8 * Tau) and (max < nmax - 1 - 0.8 * Tau) then
begin
  for i := max downto max - round(0.75 * Tau) do
    if Itrans^[i] < Imin then Imin := Itrans^[i];
  Imin1 := Imax;
  for i := max to max + round(0.75 * Tau) do
    if Itrans^[i] < Imin1 then Imin1 := Itrans^[i];
  Imin := (Imin + Imin1) / 2;
end
else
begin
  if max > 0.8 * Tau then
  begin
    for i := max downto max - round(0.75 * Tau) do
      if Itrans^[i] < Imin then Imin := Itrans^[i];
  end
  else
    for i := max to max + round(0.75 * Tau) do
      if Itrans^[i] < Imin then Imin := Itrans^[i];
end;

Imax := Imax - I0;
Imin := Imin - i0;

{ Modulation }
M := (Imax - Imin) / (Imax + Imin);

{ Kohaerenzgrad }
gam := M * (Iref^[max] + Iobj^[max]) /
  (2 * sqrt(Iref^[max] * Iobj^[max]));

{ Mittelwert ueber die Messungen }
w[j].gamma :=
  (gam + Messung * w[j].gamma) / (Messung + 1);

{ Standardabweichung }
Summe := Summe + gam;
QSumme := QSumme + sqr(gam);
if Messung > 0 then
```

```
        w[j].SAM := sqrt(abs((QSumme - 2 * Summe * w[j].gamma
          + (Messung + 1) * sqr(w[j].gamma)) / (Messung + 1)
          * Messung))
      else w[j].SAM := 0;
    end;
    if keypressed then
    begin
      Taste := readkey;
      if Taste = chr(27) then Abbruch := true;
    end;
    Dispose(DD, Done);
    if Abbruch then j := Anzahldelta - 1;
  end;
  FileName := fnsave;
  Dispose(Iref);
  Dispose(Iobj);
end; { gammaBerechnung }

{--------------------------------------------------------------------}

{ Berechnung der Kohaerenzlaenge }
procedure LCBerechnung;

var
  i                         : word;
  dc, gam, Summe, tmp, tmpa : real;

begin

  { Kohaerenzlaenge }
  tmp := 1;
  for i := 0 to Anzahldelta - 1 do
    if abs(w[i].gamma - exp(0 - 1)) < tmp then
    begin
      deltac := w[i].delta;
      tmp := abs(w[i].gamma - exp(0 - 1));
    end;

  Summe := 0;
  for i := 0 to Anzahldelta - 1 do
  begin
    gam := exp(0 - sqr(w[i].delta / deltac));
    Summe := Summe + gam - w[i].gamma;
  end;
```

```
  tmp := Summe / Anzahldelta;
  dc := deltac / 1000;

  repeat
    deltac := deltac + dc;
    tmpa := tmp;
    Summe := 0;
    for i := 0 to Anzahldelta - 1 do
    begin
      gam := exp(0 - sqr(w[i].delta / deltac));
      Summe := Summe + gam - w[i].gamma;
    end;
    tmp := Summe / Anzahldelta;
  until abs(tmp) > abs(tmpa);
  deltac := deltac - dc;

  repeat
    deltac := deltac - dc;
    tmpa := tmp;
    Summe := 0;
    for i := 0 to Anzahldelta - 1 do
    begin
      gam := exp(0 - sqr(w[i].delta / deltac));
      Summe := Summe + gam - w[i].gamma;
    end;
    tmp := Summe / Anzahldelta;
  until abs(tmp) > abs(tmpa);
  deltac := deltac + dc;

  { Standardabweichung }
  Summe := 0;
  for i := 0 to Anzahldelta - 1 do
  begin
    tmp := w[i].SAM / w[i].gamma;
    Summe := Summe + tmp;
  end;
  deltacSAM := deltac * Summe / Anzahldelta;
end; { LCBerechnung }

{-------------------------------------------------------------------}

end. { KoherU }

{-------------------------------------------------------------------}
```

D.10 Unit Anzeige

```
{-------------------------------------------------------------------}

{ Grafische Ausgabe der Messergebnisse }
unit Anzeige;

{-------------------------------------------------------------------}

interface

procedure GrafikAusgabe;
procedure LCDarstellung;

{-------------------------------------------------------------------}

implementation

uses
  ConvertU,    { Konvertierung von Plot-Daten }
  Glob,        { Globale Deklarationen }
  KoherU,      { Berechnungen Kohaerenzgrad und -laenge }
  LoadSave,    { Laden und Speichern von Dateien }
  TransU,      { Vorbereitungen zur (inversen) FFT }

  { Turbo Pascal Standard Units }
  App, Crt, Dialogs, Dos, Graph, GraphApp, MsgBox, Objects, StdDlg,
  Views;

var
  s1, s2 : string;

{-------------------------------------------------------------------}

{ Initialisierung des Grafikbildschirms }
procedure STDInitGraph;

var
  BGIPath : PString;
  GraphOk : boolean;

begin { Grafik }
  BGIPath := NewStr(FExpand('.'));
  GraphOk := GraphAppInit(0, 0, BGIPath, true);
```

```
  if not GraphOk then
    MessageBox(#3+'Kann Grafiktreiber nicht finden', nil,
      mfOKButton or mfError);
end; { STDInitGraph }

{-------------------------------------------------------------------}

{ Anlegen eines Grafikbilschirms mit festen Elementen }
procedure Grafikbildschirm;

begin
  Rectangle(0, 0, GetMaxX, GetMaxY);
  Line(0, GetMaxY - 20, GetMaxX, GetMaxY - 20);
  Line(GetMaxX - 155, GetMaxY - 20, GetMaxX - 155, GetMaxY);
  OutTextXY(15, GetMaxY - 13, s1);
  OutTextXY(GetMaxX - 140, GetMaxY - 13, s2);
end; { Grafikbildschirm }

{-------------------------------------------------------------------}

{ Zeichnen des Koordinatensystems }
procedure Koordinatenkreuz;

begin
  Line(20, GetMaxY - 40, GetMaxX - 20, GetMaxY - 40);
  Line(20, 20, 20, GetMaxY - 40);
end; { Koordinatenkreuz }

{-------------------------------------------------------------------}

{ Skalierung des Koordinatensystems fuer Ausgabe der
  Interferenzmuster und der FFT }
procedure Skalierung;

var
  i, FaktorX, FaktorY : word;

begin
  FaktorX := (GetMaxX - 39) div 10;
  FaktorY := (GetMaxY - 59) div 10;
  for i := 0 to 10 do
  begin
    Line(17, 19 + i * FaktorY, 23, 19 + i * FaktorY);
    Line(20 + i * FaktorX, GetMaxY - 37,
```

```
        20 + i * FaktorX, GetMaxY - 43);
  end;
end; { Skalierung }

{-------------------------------------------------------------------}

{ Skalierung des Koordinatensystems fuer Ausgabe der
  Kohaerenzfunktion }
procedure SkalierungLC;

var
  i, Xmax, Xskal, FaktorY : word;
  FaktorX                 : real;
  str1, str2              : string;

begin
  FaktorY := (GetMaxY - 59) div 10;
  for i := 0 to 10 do
    Line(17, 19 + i * FaktorY, 23, 19 + i * FaktorY);
  OutTextXY(8, 17, '1');
  OutTextXY(8, GetMaxY - 43, '0');
  OutTextXY(8, round(11 + 10 * (1 - exp(-1)) * FaktorY), '1');
  OutTextXY(8, round(17 + 10 * (1 - exp(-1)) * FaktorY), '-');
  OutTextXY(8, round(22 + 10 * (1 - exp(-1)) * FaktorY), 'e');
  for i := 0 to 50 do
    PutPixel(round(20 + i * (GetMaxX - 40) / 50),
             round(19 + 10 * (1 - exp(-1)) * FaktorY), white);

  Xmax := round(w[0].delta);
  for i := 1 to Anzahldelta - 1 do
    if w[i].delta > Xmax then Xmax := round(w[i].delta);
  case Xmax of
    0..10  : Xskal := 1;
    11..20 : Xskal := 2;
    21..50 : Xskal := 5;
    else Xskal := 10;
  end;
  FaktorX := (GetMaxX - 39) / XMax;
  i := 0;
  while i * XSkal <= Xmax do
  begin
    Line(round(20 + i * Xskal * FaktorX), GetMaxY - 37,
         round(20 + i * Xskal * FaktorX), GetMaxY - 43);
    str(i * Xskal, str1);
```

```
    OutTextXY(round(16 + i * Xskal * FaktorX), GetMaxY - 33, str1);
    inc(i);
  end;

  str(deltac : 3 : 1, str1);
  str(deltacSAM : 3 : 1, str2);
  if AnzahlMessung > 1 then
    str1 := 'Lc = ' + str1 + ' ' + chr(241) + ' ' + str2 + ' cm'
  else str1 := 'Lc = ' + str1 + ' cm';
  OutTextXY(GetMaxX - 180, 30, str1);
end; { SkalierungLC }

{--------------------------------------------------------------------}

{ Zeichnen einer gemessenen Intensitaetsverteilung }
procedure Intensitaetskurve;

var
  i                : word;
  FaktorX, FaktorY : real;

begin
  FaktorX := (GetMaxX - 40) / nmax;
  FaktorY := (GetMaxY - 60) / hmax;
  for i := 1 to nmax - 1 do
    Line(round((i - 1) * FaktorX + 20),
         round((hmax - 1 - Imes^[i - 1]) * FaktorY + 20),
         round(i * FaktorX + 20),
         round((hmax - 1 - Imes^[i]) * FaktorY + 20));
end; { Intensitaetskurve }

{--------------------------------------------------------------------}

{ Zeichnen einer FFT }
procedure Transformationskurve;

var
  i                      : word;
  FaktorX, FaktorY, ymax : real;

begin
  FaktorX := (GetMaxX - 40) / (nmax div 2);
  ymax := Itrans^[0];
  for i := 1 to nmax div 2 - 1 do
```

```
    if Itrans^[i] > ymax then ymax := Itrans^[i];
  FaktorY := (GetMaxY - 60) / ymax;
  for i := 1 to nmax div 2 do
    Line(round((i - 1) * FaktorX + 20),
         round((ymax - Itrans^[i - 1]) * FaktorY + 20),
         round(i * FaktorX + 20),
         round((ymax - Itrans^[i]) * FaktorY + 20));
end; { Transformationskurve }

{---------------------------------------------------------------------}

{ Zeichnen einer bereinigten Intensitaetsverteilung }
procedure Synthesekurve;

var
  i                : word;
  FaktorX, FaktorY : real;

begin
  FaktorX := (GetMaxX - 40) / nmax;
  FaktorY := (GetMaxY - 60) / hmax;
  for i := 1 to nmax - 1 do
    Line(round((i - 1) * FaktorX + 20),
         round((hmax - 1 - Itrans^[i - 1]) * FaktorY + 20),
         round(i * FaktorX + 20),
         round((hmax - 1 - Itrans^[i]) * FaktorY + 20));
end; { Synthesekurve }

{---------------------------------------------------------------------}

{ Vorbereitung zur Ausgabe einer gemessenen Intensitaetsverteilung }
procedure AusgabeMessung;

var
  Taste : char;

begin
  if not Exists(copy(FileName, 0, Length(FileName) - 3) + ExtMess)
    then Convert
  else FileName := copy(FileName, 0, Length(FileName) - 3) + ExtMess;
  LoadFileI(Imes^, FileName);
  STDInitGraph;
  if GraphicsStart then
  begin
```

```
    s1 := FileName;
    s2 := 'Weiter mit <ESC>';
    Grafikbildschirm;
    Koordinatenkreuz;
    Skalierung;
    Intensitaetskurve;
    repeat
      Taste := readkey
    until Taste in [chr(27)]; { Escape-Taste }
    GraphicsStop;
  end;
end; { Ausgabe Messung }

{-------------------------------------------------------------------}

{ Vorbereitung zur Ausgabe einer FFT }
procedure AusgabeFFT;

var
  Taste : char;

begin
  if not Exists(copy(FileName, 0, Length(FileName) - 3) + ExtFFT)
    then FFTBerechnung
  else
  begin
    FileName := copy(FileName, 0, Length(FileName) - 3) + ExtFFT;
    LoadFileR(Itrans^, FileName);
  end;
  STDInitGraph;
  if GraphicsStart then
  begin
    s1 := FileName;
    s2 := 'Weiter mit <ESC>';
    Grafikbildschirm;
    Koordinatenkreuz;
    Skalierung;
    Transformationskurve;
    repeat
      Taste := readkey
    until Taste in [chr(27)]; { Escape-Taste }
    GraphicsStop;
  end;
end; { AusgabeFFT }
```

```
{-----------------------------------------------------------------------}

{ Vorbereitung zur Ausgabe einer bereinigten Intensitaetsverteilung }
procedure AusgabeSynthese;

var
  Taste : char;

begin
  if not Exists(copy(FileName, 0, Length(FileName) - 3) + ExtKor)
    then Synthese
  else
  begin
    FileName := copy(FileName, 0, Length(FileName) - 3) + ExtKor;
    LoadFileR(Itrans^, FileName);
  end;
  STDInitGraph;
  if GraphicsStart then
  begin
    s1 := FileName;
    s2 := 'Weiter mit <ESC>';
    Grafikbildschirm;
    Koordinatenkreuz;
    Skalierung;
    Synthesekurve;
    repeat
      Taste := readkey
    until Taste in [chr(27)]; { Escape-Taste }
    GraphicsStop;
  end;
end; { AusgabeSynthese }

{-----------------------------------------------------------------------}

{ Dialog zur Ausgabe der Intensitaetsverteilungen und der FFT }
procedure GrafikAusgabe;

var
  R                           : TRect;
  D                           : PFileDialog;
  DD, DDD                     : PDialog;
  Bruce                       : PView;
  Command, Command1, Command2 : word;
```

```
begin
  Extens := '*.' + ExtPlot;
  repeat
    D := New(PFileDialog, Init(Extens, 'Datei oeffnen',
      '~N~ame (*.PLT,*.MES,*.FFT,*.KOR)',
      fdOpenButton, fdReplaceButton));
    Command := Desktop^.ExecView(D);
    if (Command <> cmCancel) then
    begin
      D^.GetFileName(FileName);
      if Exists(FileName) then
      repeat

        { Fenster der AusgabeSchalter }
        R.Assign(20, 5, 60, 17);
        DD := New(PDialog, Init(R, 'Ausgabe'));
        DD^.HelpCtx := hcAusgabe;
        with DD^ do
        begin
          R.Assign(7, 4, 32, 7);
          Bruce := New(PRadioButtons,Init(R,
            NewSItem('Messung',
            NewSItem('FFT',
            NewSItem('gefilterte Messung' ,nil)))));
          R.Assign(6, 2, 7 + Length(FileName), 3);
          Insert(New(PLabel,Init(R, FileName, Bruce)));
          Insert(Bruce);

          { Buttons }
          R.Assign(6, 9, 16, 11);
          Insert(New(PButton, Init(R, '~B~ild', cmOk, bfDefault)));
          R.Assign(24, 9, 34, 11);
          Insert(New(PButton, Init(R, '~C~ancel', cmCancel,
            bfNormal)));

          { Voreinstellung }
          SetData(AnzeigeSchalter);

          Command1 := DeskTop^.ExecView(DD);
          if Command1 <> cmCancel then
          begin

            { Parameter einlesen }
```

```
          GetData(AnzeigeSchalter);
          case Anzeigeschalter of
            0 : AusgabeMessung;
            1 : AusgabeFFT;
            2 : AusgabeSynthese;
          end;

          if AnzeigeSchalter <> 0 then
          begin
            { Option zum Speichern der Daten }
            R.Assign(0, 0, 45, 9);
            DDD := New(PDialog, Init(R,'Speichern ?'));
            with DDD^ do
            begin
              Options := Options or ofCentered;
              R.Grow(-1, -1);
              Dec(R.B.Y, 3);
              Insert(New(PStaticText, Init(R, #13
                + ^C'Daten speichern ?')));
              R.Assign(9, 6, 19, 8);
              Insert(New(PButton, Init(R, '~J~a', cmYes,
                bfDefault)));
              R.Assign(25, 6, 35, 8);
              Insert(New(PButton, Init(R, '~N~ein', cmNo,
                bfNormal)));
            end;
            Command2 := Desktop^.ExecView(DDD);
            if (Command2 = cmYes) then
              SaveFileR(ITrans^, FileName);
            Dispose(DDD, Done);
          end;
        end;
      end;
      Dispose(DD, Done);
    until Command1 = cmCancel
    else MessageBox(#3 + 'Datei ' + FileName + ' nicht gefunden' +
      #13 + #13, nil, mfError or mfOKButton);
  end;
  Dispose(D, Done);
 until Command = cmCancel;
end; { GrafikAusgabe }

{-------------------------------------------------------------------}
```

```
{ Ausgabe einer Kohaerenzfunktion }
procedure LCDarstellung;

var
  D                 : PFileDialog;
  i, Command        : word;
  FaktorX, FaktorY  : real;
  Taste             : char;

begin
  Extens := '*.' + ExtDaten;
  repeat
    D := New(PFileDialog, Init(Extens, 'Protokollatei oeffnen',
      '~N~ame (*.DAT)', fdOpenButton, fdReplaceButton));
    Command := Desktop^.ExecView(D);
    if (Command <> cmCancel) then
    begin
      D^.GetFileName(FileName);
      if Exists(FileName) then
      begin
        LoadDatFile;
        FileName :=
          copy(FileName, 0, Length(FileName) - 3) + ExtGamma;
        if not Exists(FileName) then
        begin
          Abbruch := false;
          gammaBerechnung;
          if not Abbruch then
          begin
            LCBerechnung;
            SaveGammaFile;
          end;
        end
        else LoadGammaFile;
        if not Abbruch then
        begin
          STDInitGraph;
          if GraphicsStart then
          begin
            s1 := FileName;
            s2 := 'Weiter mit <ESC>';
            Grafikbildschirm;
            Koordinatenkreuz;
            SkalierungLC;
```

```
      FaktorX := w[0].delta;
      for i := 1 to Anzahldelta - 1 do
        if w[i].delta > FaktorX then FaktorX := w[i].delta;
      FaktorX := (GetMaxX - 40) / FaktorX;
      FaktorY := (GetMaxY - 60);

      for i := 0 to Anzahldelta - 1 do
      begin
        Circle(round(w[i].delta * FaktorX + 20),
               round((1 - w[i].gamma) * FaktorY + 20), 2);
        Line(round(w[i].delta * FaktorX + 20),
             round((1 - w[i].gamma + w[i].SAM) * FaktorY + 20),
             round(w[i].delta * FaktorX + 20),
             round((1 - w[i].gamma - w[i].SAM) * FaktorY
               + 20));
        Line(round(w[i].delta * FaktorX - 2 + 20),
             round((1 - w[i].gamma + w[i].SAM) * FaktorY + 20),
             round(w[i].delta * FaktorX + 2 + 20),
             round((1 - w[i].gamma + w[i].SAM) * FaktorY
               + 20));
        Line(round(w[i].delta * FaktorX - 2 + 20),
             round((1 - w[i].gamma - w[i].SAM) * FaktorY + 20),
             round(w[i].delta * FaktorX + 2 + 20),
             round((1 - w[i].gamma - w[i].SAM) * FaktorY
               + 20));
      end;
      for i := 1 to GetMaxX - 40 do
        Line(i - 1 + 20,
             round((1 - exp(0 - (sqr((i - 1)
               / (FaktorX * deltac))))) * FaktorY + 20),
             i + 20,
             round((1 - exp(0 - (sqr(i / (FaktorX * deltac)))))
               * FaktorY + 20));
      repeat
        Taste := readkey
      until Taste in [chr(27)]; { Escape-Taste }
      GraphicsStop;
    end;
  end;
end
else MessageBox(#3 + 'Datei ' + FileName + ' nicht gefunden'
  + #13 + #13, nil, mfError or mfOKButton);
end;
Dispose(D, Done);
```

```
  until Command = cmCancel;
end; { CMDarstellung }

{-------------------------------------------------------------------}

end. { Anzeige }

{-------------------------------------------------------------------}
```

D.11 Unit LoadSave

```
{-------------------------------------------------------------------}

{ Routinen zum Laden und Speichern von Dateien }
unit LoadSave;

{-------------------------------------------------------------------}

interface

uses
  Glob;        { Globale Deklarationen }

procedure LoadFileI(var Wert : Wordarray; fn : string);
procedure SaveFileI(var Wert : Wordarray; fn : string);
procedure LoadFileR(var Wert : Realarray; fn : string);
procedure SaveFileR(var Wert : Realarray; fn : string);
procedure LoadDatFile;
procedure SaveDatFile;
procedure LoadGammaFile;
procedure SaveGammaFile;

{-------------------------------------------------------------------}

implementation

{-------------------------------------------------------------------}

{ Laden einer Integer-Messdaten-Datei }
procedure LoadFileI(var Wert : Wordarray; fn : string);

var
  i : word;
```

```
  f : Text;

begin
  Assign(f, fn);
  Reset(f);
  for i := 0 to nmax - 1 do ReadLn(f, Wert[i]);
  Close(f);
end; { LoadFileI }

{---------------------------------------------------------------------}

{ Speichern einer Integer-Messdaten-Datei }
procedure SaveFileI(var Wert : Wordarray; fn : string);

var
  i : word;
  f : Text;

begin
  Assign(f, fn);
  ReWrite(f);
  for i := 0 to nmax - 1 do WriteLn(f, Wert[i] : 4);
  Close(f);
end; { SaveFileI }

{---------------------------------------------------------------------}

{ Laden einer Real-Messdaten-Datei }
procedure LoadFileR(var Wert : Realarray; fn : string);

var
  i : word;
  f : Text;

begin
  Assign(f, fn);
  Reset(f);
  for i := 0 to nmax - 1 do ReadLn(f, Wert[i]);
  Close(f);
end; { LoadFileR }

{---------------------------------------------------------------------}

{ Speichern einer Real-Messdaten-Datei }
```

```
procedure SaveFileR(var Wert : Realarray; fn : string);

var
  i : word;
  f : Text;

begin
  Assign(f, fn);
  ReWrite(f);
  for i := 0 to nmax - 1 do WriteLn(f, Wert[i] : 9 : 3);
  Close(f);
end; { SaveFileR }

{-------------------------------------------------------------------}

{ Laden einer Protokoll-Datei }
procedure LoadDatFile;

var
  i : word;
  f : Text;

begin
  Assign(f, filename);
  Reset(f);
  ReadLn(f, Anzahldelta);
  ReadLn(f, AnzahlMessung);
  for i := 0 to Anzahldelta - 1 do
  begin
    ReadLn(f);
    ReadLn(f, df[i].dark);
    ReadLn(f, df[i].ref);
    ReadLn(f, df[i].obj);
    ReadLn(f, df[i].int);
    ReadLn(f, w[i].delta);
  end;
  Close(f);
end; { LoadDatFile }

{-------------------------------------------------------------------}

{ Speichern einer Protokoll-Datei }
procedure SaveDatFile;
```

```
var
  i : word;
  f : Text;

begin
  Assign(f, filename);
  ReWrite(f);
  WriteLn(f, Anzahldelta);
  WriteLn(f, AnzahlMessung);
  for i := 0 to Anzahldelta - 1 do
  begin
    WriteLn(f);
    WriteLn(f, df[i].dark);
    WriteLn(f, df[i].ref);
    WriteLn(f, df[i].obj);
    WriteLn(f, df[i].int);
    WriteLn(f, w[i].delta : 9 : 3);
  end;
  Close(f);
end; { SaveDatFile }

{-------------------------------------------------------------------}

{ Laden einer Ergebnis-Datei }
procedure LoadGammaFile;

var
  i : word;
  f : Text;

begin
  Assign(f, filename);
  Reset(f);
  ReadLn(f, deltac);
  ReadLn(f, deltacSAM);
  for i := 0 to Anzahldelta - 1 do
  begin
    ReadLn(f);
    ReadLn(f, w[i].delta);
    ReadLn(f, w[i].gamma);
    ReadLn(f, w[i].SAM);
  end;
  Close(f);
end; { LoadGammaFile }
```

```
{-----------------------------------------------------------------}

{ Speichern einer Ergebnis-Datei }
procedure SaveGammaFile;

var
  i : word;
  f : Text;

begin
  Assign(f, filename);
  ReWrite(f);
  WriteLn(f, deltac : 9 : 3);
  WriteLn(f, deltacSAM : 9 : 3);
  for i := 0 to Anzahldelta - 1 do
  begin
    WriteLn(f);
    WriteLn(f, w[i].delta : 9 : 3);
    WriteLn(f, w[i].gamma : 9 : 3);
    WriteLn(f, w[i].SAM : 9 : 3);
  end;
  Close(f);
end; { SaveGammaFile }

{-----------------------------------------------------------------}

end. { LoadSave }

{-----------------------------------------------------------------}
```

D.12 Unit MotorU

```
{-----------------------------------------------------------------}

{ Motorsteuerung }
unit MotorU;

{-----------------------------------------------------------------}

interface

procedure Motorsteuerung;
```

```
{----------------------------------------------------------------------}

implementation

uses
  { Turbo Pascal Standard Units }
  App, Crt, Dialogs, Objects, Views;

const
  port1a   = $1B0;       { Schreib-Lesepuffer CN1 Kanal A }
  port1b   = $1B1;       { Schreib-Lesepuffer CN1 Kanal B }
  port1c   = $1B2;       { Schreib-Lesepuffer CN1 Kanal C }
  port1r   = $1B3;       { Kontroll-Register CN1 }
  port2a   = $1B4;       { Schreib-Lesepuffer CN2 Kanal A }
  port2b   = $1B5;       { Schreib-Lesepuffer CN2 Kanal B }
  port2c   = $1B6;       { Schreib-Lesepuffer CN2 Kanal C }
  port2r   = $1B7;       { Kontroll-Register CN2 }

  timeout1 = 100;        { 100 ms Einzelschritt Vor }
  timeout2 = 10;         {  10 ms Einzelschritt Zurueck }

var
  stop : Byte;           { Zustand des Endabschalters }

{----------------------------------------------------------------------}

{ Initialisierung }
procedure Init;

begin
  port[port1r] := $92;   { CN1 Kanal A & B Eingang, Kanal C Ausgang }
  port[port2r] := $92;   { CN2 Kanal A & B Eingang, Kanal C Ausgang }
end;

{----------------------------------------------------------------------}

{ Motorvorwaertslauf einschalten }
procedure MotorVor;

begin
  port[port1c] := $40;   { 0100 0000 }
end; { MotorVor}
```

```
{-------------------------------------------------------------------}

{ Motorruecklauf einschalten }
procedure MotorZurueck;

begin
  port[port1c] := $80;   { 1000 0000 }
end; { MotorZurueck }

{-------------------------------------------------------------------}

{ Motor abschalten }
procedure MotorStop;

begin
  port[port1c] := $00;   { alle Eingaenge null }
end; { MotorStop }

{-------------------------------------------------------------------}

{ Abfrage Endabschalter }
procedure Stopsignal;

begin
  stop := port[port1b];
end; { Stopsignal }

{-------------------------------------------------------------------}

{ Dialog zur Motorsteuerung }
procedure Motorsteuerung;

var
  R     : TRect;
  D     : PDialog;
  Taste : char;

begin
  Init;
  R.Assign(0, 0, 40, 13);
  D := New(PDialog, Init(R, 'Motorsteuerung'));
  with D^ do
  begin
    Options := Options or ofCentered;
```

```
    R.Grow(-1, -1);
    Dec(R.B.Y, 3);
    Insert(New(PStaticText, Init(R, #13 + #3
      + 'vorwaerts         rueckwaerts' + #13 + #3
      + '<v>               <r>' + #13 +#3
      + 'Stop durch beliebige Taste' + #13 + #13 + #3
      + '<-    Einzelschritte    ->' + #13 + #13 + #3
      + 'Ende mit <ESC>')));
  end;
  Desktop^.Insert(D);
  repeat
    repeat
      if keypressed then
      begin
        Taste := readkey;
        case Taste of
          'V', 'v' : begin        { Motor vorwaerts }
                       MotorStop;
                       MotorVor;
                     end;
          'R', 'r' : begin        { Motor vorwaerts }
                       MotorStop;
                       MotorZurueck;
                     end;
          chr(75)  : begin        { Motor vorwaerts Einzelschritt }
                       MotorStop;
                       MotorVor;
                       delay(timeout1);
                       MotorStop;
                     end;
          chr(77)  : begin        { Motor rueckwaerts Einzelschritt }
                       MotorStop;
                       MotorZurueck;
                       delay(timeout2);
                       MotorStop;
                     end;
          else MotorStop;
        end;
      end;
      Stopsignal;
    until (stop = 1) or (stop = 2) or (Taste in [chr(27)]);
    MotorStop;
  until Taste in [chr(27)]; { Escape-Taste }
  Dispose(D, Done);
```

```
end; { Motorsteuerung }

{--------------------------------------------------------------------}

end. { MotorU }

{--------------------------------------------------------------------}
```

D.13 Unit Etc

```
{--------------------------------------------------------------------}

{ zusaetzlichen Programmteile }
unit Etc;

{--------------------------------------------------------------------}

interface

uses
  Glob,        { Globale Deklarationen }

  { Turbo Pascal Standard Units }
  App, Dialogs, Calc, Objects, Views;

procedure Rechner;
procedure Info;

{--------------------------------------------------------------------}

implementation

{--------------------------------------------------------------------}

{ einfacher Taschenrechner aus Turbo Pascal Standard Unit }
procedure Rechner;

var
  P : PCalculator;

begin
  P := New(PCalculator, Init);
  P^.HelpCtx := hcRechner;
```

```
  Desktop^.Insert(P);
end; { Rechner }

{---------------------------------------------------------------------}

{ Programm-Information }
procedure Info;

var
  D : PDialog;
  R : TRect;

begin
  R.Assign(0, 0, 45, 11);
  D := New(PDialog, Init(R,'Info'));
  with D^ do
  begin
    Options := Options or ofCentered;
    R.Grow(-1, -1);
    Dec(R.B.Y, 3);
    Insert(New(PStaticText, Init(R,
      #13 + ^C'Programm zur Messung der'#13 +
      ^C'Kohaerenzlaenge von Laserstrahlung'#13 +
      #13 + ^C'1997'#13 + ^C'Udo Becker')));
    R.Assign(17, 8, 27, 10);
    Insert(New(PButton, Init(R, '~O~K', cmOk, bfDefault)));
  end;
  Desktop^.ExecView(D);
  Dispose(D, Done);
end; { Info }

{---------------------------------------------------------------------}

end. { Etc }

{---------------------------------------------------------------------}
```

Literaturverzeichnis

[1] Eugene Hecht
Optik
Addison-Wesley Publishing Company, Inc., 1991

[2] F. und L. Pedrotti, Werner Bausch, Hartmut Schmidt
Optik, Eine Einführung
Pretice Hall, 1996

[3] Robert Guenther
Modern Optics
John Wiley & Sons, Inc., 1990

[4] Wolfgang Stößel
Fourieroptik, Eine Einführung
Springer-Verlag, 1993

[5] Jeff Hecht
The Laser Guidebook
McGraw Hill Book Company, 1986

[6] Marvin J. Weber
Handbook of Laser Sience and Technology, Supplement 1: Lasers
CRC Press, Inc., 1991

[7] Gisela Engeln-Müllges, Fritz Reutter
Formelsammlung zur Numerischen Mathematik mit Turbo-Pascal-Programmen
BI-Wissenschafts-Verlag. 1987

[8] Ralf Lindner
Aufbau eines Interferometrischen Längenmeßplatzes
Diplomarbeit, Fachhochschule Wedel, Fachbereich Physikalische Technik, 1992

[9] Stephan Schulz
Meßplatz zur Charakterisierung von Laserdioden

Diplomarbeit, Fachhochschule Wedel, Fachbereich Physikalische Technik, 1988

[10] Thomas Dresel
Grundlagen und Grenzen der 3D-Datengewinnung mit dem Kohärenzrader
Diplomarbeit, Physikalisches Institut der Universität Erlangen - Nürnberg, Lehrstuhl für Angewandte Optik, 1991

[11] Holger Venzke
Kohärenz-Rader: Ein aperturunabhngiges Verfahren zur 3D-Formerfassung optisch rauher Objekte
Diplomarbeit, Physikalisches Institut der Universität Erlangen - Nürnberg, Lehrstuhl für Angewandte Optik, 1991

[12] Ahmed Abou-Zeid, Peter Wiese
Kohärenzlängen von Laserdioden
F & M, Carl Hanser Verlag, 1995

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides Statt, daß ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form in keiner anderen Prüfungskommision vorgelegt und auch nicht veröffentlicht.

Wedel, 26. August 1997

Programmdiskette

Die Programmdiskette enhält folgende Dateien:

Im Hauptverzeichnis \

CM.exe	Meßprogramm
EGAVGA.bgi	Borland Graphics Interface

Im Unterverzeichnis **SOURCE**\

CM.pas	Hauptprogramm
Glob.pas	Globale Deklarationen
MessU.pas	Meßdialog und Einlesen der Daten
ASync4U1.pas	Schnittstellenansteuerung
ConvertU	Konvertierung der Plot-Daten
TransU.pas	Bereitstellung der Daten zur (inversen) FFT
FFTU.pas	Berechnung der (inversen) FFT ohne Coprozessor
XFFT.pas	Berechnung der (inversen) FFT mit Coprozessor
KoherU.pas	Berechnung des Kohärengrades und der Kohärenlänge
Anzeige.pas	Grafische Ausgabe der Daten
LoadSave.pas	Routinen zum Laden und Speichern
MotorU.pas	Motorsteuerung
Etc.pas	zusätzliche Programmteile

Im Unterverzeichnis **OBJECTS**\

Bitrev.obj	Object für FFTU.pas
FAsm.obj	Object für XFFT.pas